9

Grouping-Enabled and Privacy-Enhancing Communications Schemes for VANETs

T.W. Chim[1], S.M. Yiu, Lucas C.K. Hui[1] and Victor O.K. Li[2]
[1]*Department of Computer Science, The University of Hong Kong*
[2]*Department of Electrical and Electronic Engineering, The University of Hong Kong*
Hong Kong, China

1. Introduction

A vehicular ad hoc network (VANET) is also known as a vehicular sensor network [Zhang, Lu, Lin, Ho & Shen (2008)] by which driving safety is enhanced through inter-vehicle communications or communications with roadside infrastructure. It is an important element of the Intelligent Transportation Systems (ITSs) [Wang et al. (2006)]. In a typical VANET, each vehicle is assumed to have an on-board unit (OBU) and there are road-side units (RSU) installed along the roads. A trusted authority (TA) and maybe some other application servers are installed in the backend. The OBUs and RSUs communicate using the Dedicated Short Range Communications (DSRC) protocol [Oh et al. (1999)] over the wireless channel while the RSUs, TA, and the application servers communicate using a secure fixed network (e.g. the Internet). Based on this infrastructure, vehicles can broadcast safety messages (e.g. road condition, traffic accident information), referred to as "ad hoc messages", to other nearby vehicles and RSU such that other vehicles may adjust their travelling routes and RSU may inform the traffic control center to adjust traffic lights for avoiding possible traffic congestion. Like other communication networks, security issues have to be well addressed. For example, the message from an OBU has to be integrity-checked and authenticated. Otherwise, an attacker can replace the safety message from a vehicle or even impersonate a vehicle to transmit a fake safety message. For example, an attacker may impersonate an ambulance to request other vehicles to give way to it or request nearby RSUs to change traffic lights to green. Besides, privacy is another important issue. A driver may not want others to know its driving routes by tracing messages sent by its OBU. Thus an anonymous communications protocol is needed. While being anonymous, a vehicle's real identity should be revealable by a trusted party when necessary. For example, the driver who sends out fake messages causing an accident should not be able to evade responsibility by using an anonymous identity.

In terms of integrity-checking and authentication, digital signature in conventional public key infrastructure (PKI) [Housley et al. (1999)] is a well accepted choice. However, requiring a vehicle to verify the signatures of other vehicles by itself using such schemes as in [Tsang & Smith (2008)] induces two problems as mentioned in [Zhang, Lin, Lu & Ho (2008)]. First, the computation power of an OBU is not adequate to handle all verifications in a short time, especially in places where the traffic density is high. Second, to verify a message from an unknown vehicle involves the transmission of a public key certificate which causes heavy message overhead. Therefore, the general approach is to let the nearby RSU help a vehicle verify the message of another. The volume of signatures to be verified can

be huge (each vehicle is expected to broadcast a safety message every few hundred ms [U.S. Department of Transportation (2006)]). An efficient method for verifying a batch of signatures within a short period of time is desirable.
Another motivation of our work is the observation that VANET can provide a platform for a group of known vehicles (e.g. police chasing a bank robber) to establish a secure communication channel (group communications). Since communication is through a wireless channel and is more vulnerable to attacks and member vehicles would leave the group and join the group again (e.g. at junctions) rather frequently, it is desirable to have an efficient frequent group key update procedure to accommodate dynamic membership in such a group communications scheme.
To conclude, besides security and privacy requirements, an ad hoc communications protocol for VANETs should have low message overhead and efficient message verification mechanism while having high success rate and low delay. The group communications protocol should be efficient for dynamic membership as well as frequent group key update. Existing solutions cannot satisfy all these requirements. Section 2 describes related work and their limitations.
In this chapter, we propose a Grouping-enabled and Privacy-enhancing communications Scheme (or GPS in short form) for VANETs. Our schemes can handle ”ad hoc messages” (those sent by arbitrary vehicles) as well as allow vehicles that know one another in advance to form a group and send ”group messages” securely among themselves. In summary, our schemes have the following features:
1) Our schemes are software-based and do not rely on any special hardware.
2) By establishing shared secrets with RSU and TA on the handshaking phase, a vehicle is allowed to use a different pseudo identity for each session (or message) to protect its privacy while the real identity is traceable only by TA.
3) We make use of the techniques of binary search in RSU message verification phase and bloom filter in notification messages to reduce the message overhead substantially and enhance the effectiveness of the verification phase.
4) Any vehicle can form a group with other vehicles after an initial handshaking phase with a nearby RSU and then authenticate and communicate with others (either to all members or to a dedicated member) securely without the intervention of RSU even after moving into the region of another RSU.
5) We support dynamic membership in a group. When a new member
wants to join an existing group or an existing member wants to leave a group, there is no need to form a new group from scratch.
6) The group secure key can be updated periodically without any help from an RSU to increase the security level of the communication.
We provide a security analysis on our schemes and an analysis on the effectiveness of using bloom filter in the notification messages. Through analysis and simulation studies, we show that our schemes can reduce the message overhead and increase the verification success rate (will be formally defined in Section 8) by at least 45% while the additional overhead is insignificant when compared to the existing solutions.

2. State of the art

The problem of secure communications has been studied in other mobile ad hoc networks (e.g. [Kim et al. (2004)] and [Wong et al. (1998)]). However, VANET has its own characteristics which make the solutions for MANET not applicable. In brief, although mobile ad hoc network does not have a fixed topology, it may still be feasible to assume a rough topological

structure such as cluster-based, tree-based or hierarchical, and nodes in a MANET move relatively slowly. On the other hand, these assumptions are no longer valid in VANETs. In a VANET, vehicles are moving at high speed and the topology changes rapidly. There are other issues that make the problem in VANETs unique. Interested readers can refer to [Hubaux et al. (2004)] and [Raya et al. (2006)] for more details.

Ad hoc communications in VANETs have been addressed in two recent work [Zhang, Lu, Lin, Ho & Shen (2008); Zhang, Lin, Lu & Ho (2008)]. In [Zhang, Lu, Lin, Ho & Shen (2008)], the IBV protocol was proposed for vehicle-to-RSU communications. The RSU can verify a large number of signatures as a batch using just three *pairing* operations (see Section 4 for what a pairing operation is). However, their work has some limitations. First, their protocol relies heavily on a tamper-proof hardware device, installed in each vehicle, which preloads the system-wide secret key. Once one of these devices is cracked, the whole system will be compromised. Second, a vehicle's real identity can be traced by anyone, thus the protocol does not satisfy the privacy requirement. Third, their protocol has a flaw such that a vehicle can use a fake identity to avoid being traced (anti-traceability attack) or even impersonate another vehicle (impersonation attack [1]). Fourth, in their batch verification scheme, if any of the signatures is erroneous, the whole batch will be dropped. This is inefficient because most signatures in the batch may actually be valid. Finally, the IBV protocol is not designed for vehicle-to-vehicle communications.

In a more recent work [Zhang, Lin, Lu & Ho (2008)], the RAISE protocol was proposed for vehicle-to-vehicle communications. The protocol is software-based. It allows a vehicle to verify the signature of another with the aid of a nearby RSU. However, no batch verification can be done and the RSU has to verify signatures one after another. On the other hand, to notify other vehicles whether a message from a certain vehicle is valid, a hash value of 128 bytes needs to be broadcasted. There can be tens or even up to thousands of signatures within a short period of time, thus the notification messages induce a heavy message overhead.

Group communications in VANETs, on the other hand, have been considered in three papers [Chim, Yiu, Hui, Jiang & Li (2009); Wasef & Shen (2008); Verma & Huang (2009)]. In [Chim, Yiu, Hui, Jiang & Li (2009)], a scheme is proposed to allow a set of vehicles to form a group with the help of an RSU such that subsequently, encrypted group messages can be broadcasted by any member to all other members without the intervention of RSU. While group messages can be authenticated to be sent by valid members, the scheme also satisfies the privacy-preserving property that the real identity of the sender cannot be linked to the messages (except by TA) to protect the route of the vehicles being traced by unauthorized parties. However, the scheme assumes that the vehicles in the group are static in the sense that no mechanism is provided if a new member wants to join an existing group or an existing member wants to leave the group. Also, wireless channel is more vulnerable to attacks, and it is important to have an efficient scheme to update the group key periodically without the help of a third party. So, to handle dynamic membership of a group and key update based on [Chim, Yiu, Hui, Jiang & Li (2009)], we need to go through the group formation scheme from scratch with the help of an RSU.

In [Wasef & Shen (2008)], the PPGCV protocol was proposed. In addition to a scheme for group formation, they provide a protocol to update the group key. However, the setting of their scheme is different from a typical VANET and the key update process relies heavily on a key server which holds the set of all keys distributed to the vehicles.

[1] Please refer to the Appendix or [Chim, Yiu, Hui & Li (2009)] for the attacks.

In [Verma & Huang (2009)], another group communications protocol, SeGCom, was proposed. However, its concern is totally different from ours and privacy is not considered. Also RSUs are assumed to be fully trusted but this is unrealistic in practice. Furthermore, a vehicle can only form a group with all vehicles that are in the same RSU's range. It cannot select vehicles to form a group with. In all three papers, the noise of the wireless channels was not adequately addressed. We found that the success rate for forming a group degrades substantially as the noise increases in the channel.

3. Problem statement

3.1 System model and assumptions

Recall that a vehicular network consists of on-board units (OBUs) installed on vehicles, road-side units (RSUs) along the roads, and a trusted authority (TA). We focus on the inter-vehicle communications over the wireless channel. We assume the following:

1. The TA is always online and trusted. RSUs and TA communicate through a secure fixed network. To avoid being a single point of failure or a bottleneck, redundant TAs which have identical functionalities and databases are installed.
2. RSUs have higher computation power than OBUs.
3. The RSU to Vehicle Communication (RVC) range is at least twice of the Inter-Vehicle Communication (IVC) range to ensure that if an RSU receives a message, all vehicles receiving the same message are in the feasible range to receive the notification from the RSU. Consider the following counter example. Assume that the RVC range and the IVC range are both r. Two vehicles V_1 and V_2 are r apart. The distance between V_1 and RSU is within r but that between V_2 and RSU is larger than r. If V_1 sends a message to V_2, V_2 has no way to verify it as it cannot receive the notification message from the RSU. However, this problem can be resolved if the RVC range is twice that of IVC.
4. There exists a conventional public key infrastructure (PKI) for initial handshaking. The public key of TA PK_{TA} is known by *everyone*. The public key of vehicle V_i PK_{V_i} is known by TA. Also any RSU R_k broadcasts its public key PK_{R_k} with hello messages periodically to vehicles that are travelling within RVC range of it. Thus PK_R is known by all vehicles nearby. The TA and RSU R_k keep their secret keys SK_{TA} and SK_{R_k} privately. There is no need for vehicles to know the public keys of other vehicles to avoid message overhead for exchanging certificates.
5. The real identity of any vehicle is only known by TA and itself but not by others.

3.2 Security requirements

We aim at designing schemes to satisfy the following security requirements:

1 *Message integrity and authentication*: A vehicle should be able to verify that a message is indeed sent and signed by another vehicle without being modified by anyone.

2 *Identity privacy preserving*: The real identity of a vehicle should be kept anonymous from other vehicles and a third party should not be able to reveal a vehicle's real identity by analysing multiple messages sent by it.

3 *Traceability*: Although a vehicle's real identity should be hidden from other vehicles, if necessary, TA should have the ability to obtain a vehicle's real identity and relate the message to the sender (for example, in case the real identity of the sender of a fake message causing an accident needs to be revealed).

4 *Confidentiality*: Group messages broadcasted to all members should not be decryptable by vehicles not in the group and a group message sent to a dedicated member should only be decryptable by that dedicated receiver, other vehicles (including other members) should not be able to decrypt the message.

4. Preliminaries

Our schemes are *pairing-based* and defined on two cyclic groups with a *bilinear mapping* [Menezes (1991)]. We briefly introduce what a bilinear map is and will discuss the basics on bloom filter which we apply in the RSU notification phase.

4.1 Elliptic curve cryptography (ECC)

Let $\mathbb{G}$ be a cyclic additive group with generator P and $\mathbb{G}_{\mathbb{T}}$ be a cyclic multiplicative group. Both groups $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ have the same prime order q. The mapping $\hat{e} : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_{\mathbb{T}}$ is called a *bilinear map* if it satisfies the following properties:

1. Bilinear: $\forall P, Q, R \in \mathbb{G}$ and $\forall a, b \in \mathbb{Z}$, $\hat{e}(Q, P + R) = \hat{e}(P + R, Q) = \hat{e}(P, Q) \cdot \hat{e}(R, Q)$. Also $\hat{e}(aP, bP) = \hat{e}(P, bP)^a = \hat{e}(aP, P)^b = \hat{e}(P, P)^{ab}$.
2. Non-degenerate: There exists $P, Q \in \mathbb{G}$ such that $\hat{e}(P, Q) \neq 1_{\mathbb{G}_{\mathbb{T}}}$.
3. Computable: There exists an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in \mathbb{G}$.

The bilinear map $\hat{e}$ can be constructed on elliptic curves. Each operation for computing $\hat{e}(P, Q)$ is a *pairing operation*. Pairing operation is the most expensive operation in this kind of cryptographic schemes. The fewer the number of pairing operations, the more efficient the scheme is. The groups $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ are called bilinear groups. The security of our schemes relies on the fact that the discrete logarithm problem (DLP) on bilinear groups is computationally hard, i.e., given the point $Q = aP$, there exists no efficient algorithm to obtain a by given P and Q. The implication is that we can transfer Q in an open wireless channel without worrying that a (usually some secret) can be known by the attackers.

4.2 Bloom filter

A *bloom filter* is a method for representing a set $A = a_1, a_2, ..., a_n$ of n elements to support membership queries. The idea is to allocate a vector v with m bits, initially all set to 0, and then choose k independent hash functions, h_1, h_2, ..., h_k, each with range 1, ..., m. For each element $a \in A$, the bits at the positions $h_1(a)$, $h_2(a)$, ..., $h_k(a)$ in v are set to 1 (A particular bit might be set to 1 multiple times). To answer if a value b is in A, we check the bits at positions $h_1(b)$, $h_2(b)$, ..., $h_k(b)$. If any of them is 0, then b is definitely not in the set A. Otherwise we conjecture that b is in the set although there is a certain probability that we are wrong (called a false positive). After inserting n keys into the vector with m bits with k hash functions, the probability that a particular bit is still 0 is $(1 - \frac{1}{m})^{kn} \sim e^{-\frac{kn}{m}}$ assuming that on any input value, the hash functions pick each position with equal probability. Hence the probability of a false positive is $(1 - (1 - \frac{1}{m})^{kn})^k \sim (1 - e^{-\frac{kn}{m}})^k$. Let $f(k) = (1 - e^{-\frac{kn}{m}})^k$ and let $g(k) = lnf(k) = kln(1 - e^{-\frac{kn}{m}})$. By finding $\frac{dg}{dk}$ and making $\frac{dg}{dk} = 0$, it can be shown that to minimize the probability of having false positives, k should be set to $\frac{m \ln 2}{n}$.

5. Our scheme for ad hoc communications

This section presents our scheme for ad hoc communications in details. There are some initial parameters to be generated by TA using the following steps. This needs to be done once

for the whole system unless the master key, or the real identity of a vehicle are believed to be compromised, or TA wants to update the parameters and the master key periodically to enhance the security level of the system.
(1) TA chooses $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ that satisfy the bilinear map properties.
(2) TA randomly picks $s \in \mathbb{Z}_q$ as its master key and computes $P_{pub} = sP$ as its public key. The public parameters $\{\mathbb{G}, \mathbb{G}_{\mathbb{T}}, q, P, P_{pub}\}$ are publicly accessible by all RSUs and vehicles.
(3) TA assigns each vehicle V_i a real identity $RID_i \in \mathbb{G}$ and a password PWD_i. The drivers are informed about them during network deployment or during vehicle first registration.
Our scheme can be divided into the following modules:
(A) **Initial handshaking**: This module is executed when a vehicle meets a new RSU. The vehicle authenticates itself with TA via RSU. Note that TA is the only authorized party which knows the real identity of the vehicle, so TA will pass information to RSU to allow RSU to verify the vehicle's signature even if it uses pseudo identity to sign the message. Also, RSU will generate a shared secret with the vehicle. If this is the first time the vehicle authenticates itself with TA, TA will also pass its master key s and a shared secret to the vehicle. This only needs to be done once in the whole session. To increase the security level, s is not preloaded into any hardware on the vehicle as in [Zhang, Lu, Lin, Ho & Shen (2008)]. For the shared secret with RSU, a new secret is generated every time the vehicle moves into the region of another RSU.
(B) **Message signing**: When a vehicle wants to send out a message, it first creates a pseudo identity together with the signing key. This can be done *per message* to increase the difficulty of attackers attempting to trace its real identity. Then, it signs the message using the signing key of the pseudo identity.
(C) **Batch verification**: This module is used by the RSU to verify a set of messages using only *two* pairing operations in a batch mode. We also describe how to generate a notification broadcast message using bloom filter and how to handle the case in which there are some invalid signatures in the batch (recall that in [Zhang, Lu, Lin, Ho & Shen (2008)], once there is an invalid signature in the batch, the whole batch of signatures are assumed to be invalid and ignored).
(D) **Real identity tracking**: This module is used by TA to reveal the real identity of the sender of a given message.

5.1 Initial handshaking

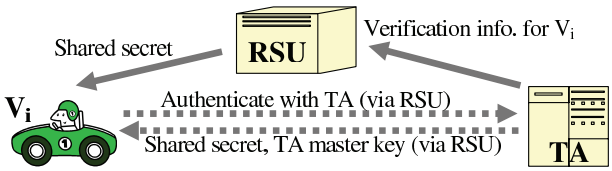

Fig. 1. Initial handshaking

We use the notations $CENC_Z(M)$, $CDEC_Z(M)$ and $CSIG_Z(M)$ to denote conventional encrypting, decrypting and signing, respectively, message M using the key Z. To enhance readability, we summarize the notations that will be used in this chapter in Table 1. The detailed processes in this module are as follows:

1. When a vehicle V_i meets the first RSU R_k, it encrypts its RID_i and PWD_i using TA's public key PK_{TA} and sends $ENC_{PK_{TA}}(RID_i, PWD_i, r)$ to the RSU which forwards it to TA. Here RID_i, PWD_i and r are concatenated in a pre-defined way and r is a random nonce. By including r, two similar blocks sent by the same vehicle cannot be related by an attacker.

Symbol	Meaning
$CENC_Z(M)$	Encrypting message M using the key Z
$CDEC_Z(M)$	Decrypting message M using the key Z
$CSIG_Z(M)$	Signing message M using the key Z
$H(.)$	MapToPoint hash function [Boneh et al. (2001)]
$h(.)$	One-way hash function such as SHA-1 [Eastlake & Jones (2001)]
r	random number
TA	Trusted authority
R_k	RSU number k
V_i	Vehicle number i
PK_{TA}	Public key of TA
SK_{TA}	Secret key of TA
PK_{R_k}	Public key of RSU R_k
SK_{R_k}	Secret key of RSU R_k
PK_{V_i}	Public key of vehicle V_i
s	TA's master key
$\{\mathbb{G}, \mathbb{G}_{\mathbb{T}}, q, P, P_{pub} = sP\}$	Public parameters
RID_i	Real identity of vehicle V_i
PWD_i	Password of vehicle V_i
t_i	Shared secret between TA and vehicle V_i
m_i	Shared secret between RSU R_k and vehicle V_i
M_i / M_r	Messages sent by vehicle / RSU
ID_i	Pseudo identity of vehicle V_i
VPK_i	Verification public key of vehicle V_i
$SK_i = (SK_{i1}, SK_{i2})$	Signing key of vehicle V_i
σ_i	ECC signature by vehicle V_i
LID_i	Local pseudo identity of vehicle V_i (for group communications)
GPK_i	Group public key of vehicle V_i (for group communications)
$LSK_i = (LSK_{i1}, LSK_{i2})$	Local signing key of vehicle V_i (for group communications)
ς_j	Local ECC signature by vehicle V_i (for group communications)
rr	Partial group secret key (for group communications)
β	Group secret key (for group communications)

Table 1. Notations used in this chapter

2. TA decrypts and verifies RID_i and PWD_i. If they are valid, it generates a shared secret t_i for V_i and computes V_i's ID Verification Public Key as $VPK_i = t_i \oplus RID_i$. TA then passes VPK_i to RSU to enable it to verify signatures from V_i even if V_i uses pseudo identity to sign the message. TA then stores the (RID_i, t_i) pair into its repository and forwards PK_{V_i}, VPK_i and $X = ENC_{PK_{V_i}}(s, VPK_i, SIG_{SK_{TA}}(s, VPK_i))$ to RSU, where PK_R and PK_{V_i} are conventional public keys of RSU and vehicle V_i, respectively. Note that to let V_i know that s and VPK_i are really sent by TA, TA includes its signature on s and VPK_i ($CSIG_{SK_{TA}}(s, VPK_i)$) into the encrypted text.
3. RSU chooses a random number m_i to be the shared secret between itself and vehicle V_i. It stores the (VPK_i, m_i) pair into its verification table for later usage. It then sends $Y = ENC_{PK_{V_i}}(m_i, CSIG_{SK_R}(m_i))$ and X to vehicle V_i. Again to let vehicle V_i know that m_i is really sent by RSU, RSU signs it.
4. Vehicle V_i decrypts Y to obtain m_i and verifies RSU's signature on it. Similarly, it decrypts X to obtain s and VPK_i and verifies TA's signature on them. It then computes its shared secret with TA using $t = VPK_i \oplus RID_i$.

This basically completes the initial handshaking phase. The following shows the procedure when vehicle V_i leaves the range of an RSU and enters the range of another. It includes a simpler authentication process with TA so that TA can pass the information to the new RSU for verifying V_i's signature and a new shared secret will be generated by this RSU.

5) V_i sends $ENC_{PK_{TA}}(RID_i)$ to TA via this new RSU. This time TA does not need to verify V_i's PWD anymore as it has already done that when V_i first starts up. Instead it directly generates a new t_i and a new VPK_i for V_i and sends VPK_i to the new RSU. TA then adds the new t_i into its repository. Next the new RSU chooses a random number m_i to be its shared secret with V_i. After storing (VPK_i, m_i) into its verification table, RSU sends $Y = ENC_{PK_{V_i}}(m_i, CSIG_{SK_R}(m_i))$ to V_i which then decrypts it using its conventional secret key. From now on, vehicle V_i starts to use the new shared secret with the new RSU for message signing.

5.2 Message signing

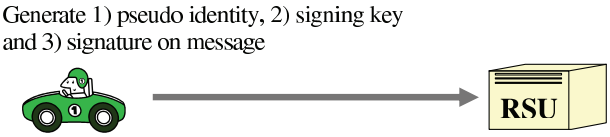

Fig. 2. Message signing

To sign a message, a vehicle generates a pseudo identity and the corresponding signing key. A different pseudo identity can be used for a different message.
To generate a pseudo identity, V_i first generates a random nonce r. Its pseudo identity ID_i contains two parts - ID_{i1} and ID_{i2} where $ID_{i1} = rP_{pub}$ and $ID_{i2} = VPK_i \oplus H(m_i ID_{i1})$. The corresponding signing key is $SK_i = (SK_{i1}, SK_{i2})$ where $SK_{i1} = sm_i ID_{i1}$ and $SK_{i2} = sH(ID_{i2})$. $H(.)$ is a MapToPoint hash function [Boneh et al. (2001)]. Then, to sign a message M_i, V_i computes the signature $\sigma_i = SK_{i1} + h(M_i)SK_{i2}$ where $h(.)$ is a one-way hash function such as SHA-1 [Eastlake & Jones (2001)]. Vehicle V_i then sends $< ID_i, M_i, \sigma_i >$ to others.

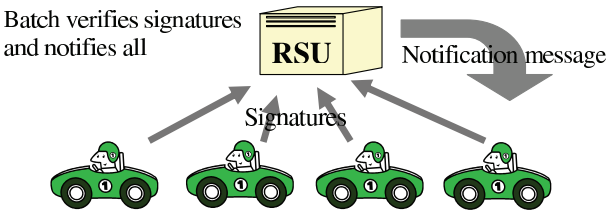

Fig. 3. Batch verification

5.3 Batch verification

This module allows an RSU to verify a batch of signatures using only two pairing operations based on the bilinear property of the bilinear map. We require an RSU to perform batch verification at a frequency higher than that with which a vehicle broadcasts safety messages so that a vehicle can verify the safety message of another before it broadcasts a more updated one. We first show the verification procedure. Then, we show how to make use of bloom filter to construct a notification message in order to reduce the message overhead. Lastly, we describe how to handle the case in which there are invalid signatures in the batch and how to extract valid ones from the batch instead of dropping the whole batch as in [Zhang, Lu, Lin, Ho & Shen (2008)].

Verification procedure. Assume that RSU wants to verify a batch of signatures σ_1, σ_2, ..., σ_n from vehicles V_1, V_2, ..., V_n on messages M_1, M_2, ..., M_n. With the shared secrets and the pseudo identities of the vehicles, RSU first determines their verification public keys $VPK_1, VPK_2, \ldots, VPK_n$ and shared secrets $m_1, m_2, \ldots, m_n$ by checking which of the stored (VPK_i, m_i) pairs satisfy $ID_{i2} = VPK_i \oplus H(m_i ID_{i1})$. It then verifies the signatures by checking if $\hat{e}(\sum_{i=1}^{n} \sigma_i, P) = \hat{e}(\sum_{i=1}^{n} m_i ID_{i1} + h(M_i)H(ID_{i2}), P_{pub})$ as

$\hat{e}(\sum_{i=1}^{n} \sigma_i, P)$
$= \hat{e}(\sum_{i=1}^{n} SK_{i1} + h(M_i)SK_{i2}, P)$
$= \hat{e}(\sum_{i=1}^{n} SK_{i1}, P)\hat{e}(\sum_{i=1}^{n} h(M_i)SK_{i2}, P)$
$= \hat{e}(\sum_{i=1}^{n} sm_i ID_{i1}, P)\hat{e}(\sum_{i=1}^{n} h(M_i)sH(ID_{i2}), P)$
$= \hat{e}(\sum_{i=1}^{n} m_i ID_{i1}, sP)\hat{e}(\sum_{i=1}^{n} h(M_i)H(ID_{i2}), sP)$
$= \hat{e}(\sum_{i=1}^{n} m_i ID_{i1}, P_{pub})\hat{e}(\sum_{i=1}^{n} h(M_i)H(ID_{i2}), P_{pub})$
$= \hat{e}(\sum_{i=1}^{n} m_i ID_{i1} + h(M_i)H(ID_{i2}), P_{pub})$.

To avoid replay attack, an RSU stores the pseudo identities used by vehicles. If the pseudo identity in a vehicle's message matches any stored one, RSU rejects the message immediately. Note that if a vehicle does not know the shared secret with RSU, it cannot produce a valid signature. There may be a very small chance that the pseudo identities generated by two vehicles are the same. In that case, RSU will treat the signatures as invalid. The vehicles will sign again using a different pseudo identity.

Generating notification message. After RSU verifies vehicle V_i's signature σ_i, it notifies all vehicles within its RVC range the result. We first assume that all signatures are valid. For each valid message, we store a hash value $h(ID_i, M_i)$ of the message in the bloom filter (the hashing function is known to everyone) to minimize message overhead. However, as we have discussed in Section 4.2, there can be false positives in a bloom filter. To reduce this impact, we propose to use two bloom filters which contain opposite information: *Positive and Negative Filter*. The positive bloom filter stores the hash value of pseudo identities and messages of vehicles whose signatures are valid and the negative bloom filter stores the hash value of pseudo identities and messages of vehicles whose signatures are invalid.

If vehicle V_i wants to verify vehicle V_j's signature σ_j on message M_j, it first computes $h(ID_i, M_i)$ and then checks the positive filter and the negative filter as included in the RSU broadcast. There are four possible cases (see Table 2). For the first two cases, the resulting validity of σ_j can be confirmed. For the third case, V_j's hash appears in both filters. Then this must be a false positive in either filter, thus a re-confirmation procedure is needed. For the last case, V_j's hash does not appear in both filters. It means that RSU still has not yet verified σ_j and so V_i has to wait for RSU's next broadcasting message.

To facilitate re-confirmation, we require a vehicle to store the signatures of other vehicles which they are interested in upon receiving them for the first time for a short period. Also we require RSU to store the valid signatures that it has verified together with the sending vehicles' pseudo identities for at least one more batch verification period after that signature is lastly requested.

If Case 3 occurs, vehicle V_i re-sends σ_j to RSU. RSU searches for σ_j from those stored signatures. If σ_j can be found, RSU adds the hash of V_j into the positive filter. Otherwise, it adds it into the negative filter. All re-confirmation results can be embedded into a re-confirmation reply similar to a normal notification message. In practice, we can use one bit to distinguish whether the reply is a normal notification message or a re-confirmation reply.

Case	Positive Filter	Negative Filter	Validity of σ_j
1	True	False	Valid
2	False	True	Invalid
3	True	True	(Re-confirmation needed)
4	False	False	(Wait for next broadcast)

Table 2. Possible cases in bloom filters and their implications

There is still a chance that Case 3 occurs again. Our scheme allows the use of bloom filters for re-confirmation for K rounds. If after K rounds and Case 3 still occurs, RSU will send $h(ID_j, M_j)$ of V_j to vehicle V_i as a direct notification. To facilitate RSU to know what it should send in the re-confirmation reply, RSU stores the number of requests to each of its signature stored. See the next section for the performance of our schemes with different values of K.

Note that the size of each bloom filter m (i.e. the number of bits used) can be a variable in our schemes to save transmission overhead. To help the receiving vehicles to determine the size of the filters (so that they can adjust the range of hash functions accordingly), together with the valid and the invalid filters, RSU also transmits a value n to represent the total number of signatures in the batch (i.e. the number of values being added into any bloom filter cannot exceed n). To allow vehicles to confirm that a notification message is indeed sent by an RSU, RSU signs the bloom filters using its private key SK_R before broadcasting them.

Invalid signatures in the batch. A batch may contain tens and even up to thousands of signatures depending on the traffic density around RSU. In the IBV protocol, if any of the signatures inside the batch is invalid, the whole batch is dropped. This approach is inefficient in the sense that most of the signatures in the batch are actually valid and can be used. Thus in our schemes, we propose to adopt binary search in the verification process to extract those valid ones. Assume that the batch contains n signatures, we arrange them in a fixed order (say according to the senders' pseudo identities). If the batch verification fails, we first determine the mid-point as $mid = \lfloor \frac{1+n}{2} \rfloor$. Then we perform batch verification on the first half (the 1^{st} to mid^{th} elements) and the second half (the $(mid+1)^{th}$ to n^{th} elements) separately. If any of the two batches causes a failure in the verification again, we repeat the same process on the invalid batch. If the pairing on any batch is valid, the RSU notifies all those signatures immediately.

The binary search stops if a batch contains only one signature or when a pre-defined level of binary search is reached. In Section 8, we evaluate the performance of our schemes using different number of levels in binary search and it is found that a full exploration may not be necessary in most cases.

5.4 Real identity tracking

To reveal the real identity of the sender of a message, TA is the only authorized party that can perform the tracing. Given vehicle V_i's pseudo identity ID_i and its shared secret with the connecting RSU m_i, TA can search through all the stored (RID_j, t_j) pairs from its repository. Vehicle V_i's real identity is the RID_j value from the entry that satisfies the expression $ID_{i2} \oplus t_j \oplus H(m_i ID_{i1}) = RID_j$ as $ID_{i2} \oplus t_j \oplus H(m_i ID_{i1}) = t_i \oplus RID_j \oplus H(m_i ID_{i1}) \oplus t_i \oplus H(m_i ID_{i1}) = RID_j$. No other party can obtain vehicle V_i's real identity since t_i is only known by TA and V_i itself.

6. Our scheme for group communications

This section presents our scheme for group communications in details. This scheme is based on the framework of our ad hoc communications scheme in Section 5. The scheme can be divided into the following modules:
(A) **Group formation**: This module is used when a set of vehicles want to form a group. A group partial secret key and a set of group public keys for group members will be generated by TA and forwarded by an RSU.
(B) **Secure one-to-many and one-to-one communications**: This module describes how a vehicle can send a message securely to all other members or to a dedicated member in the group.
(C) **New member joining**: This module is invoked when a new member wants to join an existing group.
(D) **Common group secret update**: This module shows how the common group secret can be updated without the help of RSU.
(E) **Member leaving**: This module is invoked when a member wants to leave a group.
(F) **Real identity tracking**: This module is used by TA to reveal the real identity of the sender of a given message.

6.1 Group formation

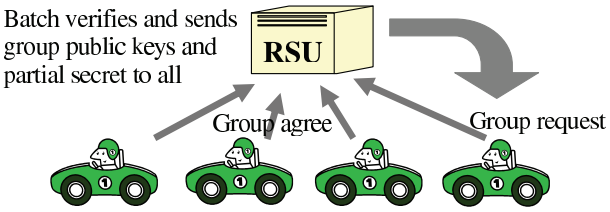

Fig. 4. Group Key Generation

When a group of vehicles want to form a group, each of them first creates a pseduo identity together with the signing key. This can be done per message to increase the difficulty of attackers to trace its real identity. Then, it signs a control message using the signing key of the pseudo identity. RSU verifies the set of control messages using only *two* pairing operations in a batch mode and distributes necessary information to vehicles in a group so that they can

verify each others' messages without the aid of any RSU later on. A partial group secret key will also be generated. Details are as follow.

Assume that vehicles V_1, V_2, ..., V_n have already registered with the closest RSU and their shared secrets with RSU are m_1, m_2, ..., m_n respectively. Also assume that these vehicles know pseudo identities of one another already.

Group request. Vehicle V_i generates the group request message $M_i = \{GPREQ, ID_1, ID_2, ..., ID_{i-1}, ID_{i+1}, ..., ID_n\}$ and its signature σ_i on M_i using the method in Section 5.2. Here ID_j for all $j \neq i$ is the pseudo identity lastly used by V_j as heard by V_i. V_i then broadcasts $< ID_i, M_i, \sigma_i >$ to RSU and others. Note that V_i can be anyone or the leader of the group.

Group agree. Any vehicle V_j receiving V_i's $GPREQ$ message checks whether its lastly used pseudo identity is included in the $GPREQ$ message. If yes, it composes the message $M_j = \{GPAGR, ID_j\}$ and its signature σ_j on it and sends $< ID_j, M_j, \sigma_j >$ to RSU. Note that vehicle V_j generates ID_j and σ_j using the method in Section 5.2 above.

Verification and key generation. At fixed intervals, RSU verifies the group request and group agree messages. Note that there is a method to batch verify a set of incoming signatures as we discussed in Section 5.3. For any vehicle V_x whose signature is found to be valid, it generates its group public key as $GPK_x = m_x P$ and stores it into the verification table. Besides, it also generates a random number rr which will be used to form the group secret key among the group of vehicles. Without loss of generality, assume the signatures from V_1, ..., V_x are valid, RSU broadcasts $M_r = \{ID_1, ID_2, ..., ID_x, GPK_1, GPK_2, ..., GPK_x, CENC_{m_1}(rr), CENC_{m_2}(rr), ..., CENC_{m_x}(rr)\}$ and its signature $CSIG_{CSK_R}(M_r)$ to the vehicles concerned.

In case the verification fails due to invalid signatures or vehicles inside the range have the same pseudo identity (although the chance is very small), RSU will stop the protocol and the group is required to repeat the protocol again for the sake of security.

Key reception and acknowledgement. Upon receiving RSU's broadcast, each vehicle V_i in the group acknowledges RSU by composing $M_i = \{KEYRECV\}$ and sending out the reply $< ID_i, M_i, \sigma_i >$. Note that ID_i and σ_i are generated in the same way as in the group request or group agree message. If after a timeout period (which is a system parameter), RSU still cannot receive the acknowledgement from V_i, it assumes that the message is corrupted on its way to V_i. More than one vehicle may not acknowledge. RSU then resends the previous broadcast to all these vehicles but this time, ID_j and $CENC_{m_j}(rr)$ of all vehicles V_j who have acknowledged do not need to be included in the broadcast anymore. In Section 8, we will show that acknowledgements are important in increasing the group formation success rate. Each vehicle in the group then stores all the group public keys and the decrypted rr values.

6.2 Secure one-to-many and one-to-one communications

In this sub-section, we describe how a vehicle can send a message securely to all other members or to a dedicated member in the group in detail. We also describe how a vehicle can sign a message so that another member can ensure the message is indeed sent by it. We consider the communication between two vehicles in a group as a *local* communication.

One-to-many communications. The vehicles in the group compute a common group secret as $\beta = s \times rr$ (note that RSU does not know how to compute β since s is only known by vehicles but not RSU) and they can communicate with each other securely using any symmetric key cryptographic algorithm such as DES [Brown et al. (1993)] from now on.

One-to-one communications. Based on the stored group public key GPK_j, when vehicle V_i wants to send a message M_i to another member V_j, it first generates a random number x. It then computes the ciphertext $C_i = \{xP, M_i + sxGPK_j\}$. To decrypt, it multiplies xP by sm_j and subtracts the result from the second part to obtain M_i as $M_i + sxGPK_j - xsm_jP = M_i + sxm_jP - xsm_jP = M_i$. We denote the encryption and decryption processes here as $C_i = EENC_{GPK_j}(M_i)$ and $M_i = EDEC_{m_j}(C_i)$, respectively.

Local message signing and verification. Next we look at the *local* pseudo identity generation, message signing and signature verification when group communications (either one-to-many or one-to-one) take place. To denote the *local* nature, we add the character L in front of the notations LID_i and LSK_i. When vehicle V_i wants to send a local message M_i, it first generates its *local* pseudo identity $LID_i = (LID_{i1}, LID_{i2})$ where $LID_{i1} = rP$ and $LID_{i2} = GPK_i + r\beta P$. r is again a random nonce here. Then it composes its *local* signing key $LSK_i = (LSK_{i1}, LSK_{i2})$ as $LSK_{i1} = sm_iLID_{i1}$ and $LSK_{i2} = sm_iH(LID_{i2})$ where $H(.)$ is a MapToPoint hash function as before. It signs the message M_i by computing the *local* signature $\varsigma_i = LSK_{i1} + h(M_i)LSK_{i2}$ where $h(.)$ is a one-way hash function (note that we use a different notation to differentiate it from a non-local signature). Finally, it sends to others $< LID_i, C_i, \varsigma_i >$ where C_i is the ciphertext corresponding to M_i.

Assume vehicle V_j wants to verify the signature of V_i on M_i. It first retrieves V_i's group public key GPK_i by computing $LID_{i2} - \beta LID_{i1}$ because $LID_{i2} - \beta LID_{i1} = GPK_i + r\beta P - \beta rP = GPK_i$. Then it decrypts C_i to obtain M_i and checks whether $\hat{e}(\varsigma_i, P) = \hat{e}(LID_{i1} + h(M_i)H(LID_{i2}), sGPK_i)$ as

$\hat{e}(\varsigma_i, P)$
$= \hat{e}(LSK_{i1} + h(M_i)LSK_{i2}, P)$
$= \hat{e}(LSK_{i1}, P)\hat{e}(h(M_i)LSK_{i2}, P)$
$= \hat{e}(sm_iLID_{i1}, P)\hat{e}(h(M_i)sm_iH(LID_{i2}), P)$
$= \hat{e}(LID_{i1}, sm_iP)\hat{e}(h(M_i)H(LID_{i2}), sm_iP)$
$= \hat{e}(LID_{i1}, sGPK_i)\hat{e}(h(M_i)H(LID_{i2}), sGPK_i)$
$= \hat{e}(LID_{i1} + h(M_i)H(LID_{i2}), sGPK_i)$.

6.3 New member joining

Assume that vehicle V_k wants to join the group of V_i, namely, $(V_1, \ldots, V_n)$ which are in the range of the same RSU and the shared secret between V_k and RSU is m_k.

Group join. V_k first composes a group join message $M_k =< GPJOIN, ID_i >$ with its signature σ_k on it. It sends $< ID_k, M_k, \sigma_k >$ to the closest RSU. Note that ID_i is the pseudo identity lastly used by V_i as seen by V_k. It is not a local pseudo identity since V_k still has not joined V_i's group yet. Also V_k generates its pseudo identity ID_k and signature σ_k in the same manner as other vehicles when they send out their group request or group agree messages.

Group join agree. When V_i finds that its last used pseudo identity is inside V_k's group join message, V_i replies with a group join agree message $< ID_i, M_i, \sigma_i >$ where ID_i is generated as usual and $M_i = \{GPJOINAGR||ID_k||CENC_{CPK_k}(rr)\}$.

Verification by RSU and key generation. Upon receiving the group join and group join agree messages from V_k and V_i, respectively, RSU verifies them. RSU then generates V_k's group public key as $GPK_k = m_kP$. It broadcasts $M_r = \{ID_k||ID_i||GPK_k||GPK_i, CENC_{m_k}(rr)\}$ and its signature $CSIG_{CSK_R}(M_r)$ to V_k and V_i.

Key reception and acknowledgement. Upon receiving RSU's broadcast, V_j where $j \in \{i,k\}$ verifies RSU's signature and acknowledges it by composing $M_j = \{KEYRECV\}$ and sending out the reply $< ID_j, M_j, \sigma_j >$. Note that ID_j and σ_j are generated in the same way as in the

group join or group join agree message. If after a timeout period, RSU still cannot receive the acknowledgement from either V_k or V_i, it resends the previous broadcast to it. V_k then decrypts $CENC_{m_k}(rr)$ using its shared secret with RSU m_k and computes the common group secret as $\beta = s \times rr$.

Sharing of group public keys. Up to this moment, only V_i knows how to verify signatures by V_k. Thus V_i shares this piece of information with other members by composing the message $M_i = \{NEWMEMBER||GPK_k\}$ and broadcasting $< LID_i, M_i, \varsigma_i >$ to other members. Each member V_j verifies the signature of V_i and acknowledges V_i with the reply message $< LID_j, M_j, \varsigma_j >$ where LID_j is the local pseudo identity of V_j and $M_j = \{GPKRECV\}$. If after a timeout period, V_i still cannot receive the acknowledgement from any member, it resends the previous broadcast to it.

After all, one task is still missing. That is to inform V_k about how to verify other members' signatures. This task is again assigned to V_i. V_i composes the message $M_i = \{GPK_1, GPK_2, ..., GPK_{i-1}, GPK_{i+1}, ..., GPK_n\}$ and sends $< LID_i, M_i, \varsigma_i >$ to V_k. Upon receiving V_i's message, V_k acknowledges it like what other members do.

6.4 Common group secret update

Now we show how to update the common group secret β periodically without the help of RSU. Each member V_i can periodically request a key update by broadcasting the message $< LID_i, M_i, \varsigma_i >$ where LID_i is the local pseudo identity of V_i and $M_i = \{CGSUPDATE\}$. The requester V_i then generates a new random number rr_{new} and computes $\beta_{new} = rr_{new} \times s$. It sends to each other member V_j the message $< LID_i, M_i, \varsigma_i >$ where LID_i is the local pseudo identity of V_i and $M_i = \{NEWCGS, EENC_{GPK_j}(\beta_{new})\}$.

Each V_j acknowledges V_i with the reply message $< LID_j, M_j, \varsigma_j >$ where LID_j is the local pseudo identity of V_j and $M_j = \{NEWCGSRECV\}$. If after a timeout period, V_i still cannot receive the acknowledgement from V_j, it resends the previous message to it. Note that the acknowledgements here ensure that all members staying in the group can receive the new common group secret properly for ongoing one-to-many communications.

6.5 Member leaving

When a member V_k wants to leave a group, the group common secret should be updated so that V_k can no longer decrypt the group's ongoing communications. We can simply conduct a group key update protocol excluding V_k.

6.6 Real identity tracking

Again only TA can trace the real identity of the sender of a message. For vehicle V_i's group request or group agree message, TA can trace V_i's real identity using the routine in Section 5.4. For vehicle V_i's local message to other members, the connecting RSU first retrieves V_i's group public key GPK_i by computing $LID_{i2} - \beta LID_{i1}$ similar to what the receiver does. Then it looks up its verification table to retrieve V_i's verification public key VPK_i which was assigned by TA. By presenting VPK_i, TA can search through all the stored (RID_j, t_j, m_j) tuples from its repository. Vehicle V_i's real identity is the RID_j value from the entry that satisfies the expression $RID_j = t_j \oplus VPK_i$.

7. Analysis

7.1 Security analysis

We analyse our schemes with respect to the security requirements listed in Section 3.

Message integrity and authentication: For ad hoc messages, the signature σ_i on message M_i by vehicle V_i is composed of SK_{i1} and SK_{i2}. SK_{i1} is defined as sm_iID_{i1} where m_i is the shared secret between vehicle V_i and the RSU. Due to the difficulty of solving the discrete logarithm problem, there is no way for attackers to reveal m_i. Thus the attacker cannot forge a signature. Similarly, for group message, although all vehicles in the group know the group public key $GPK_i = m_iP$ of V_i, it is computationally hard to obtain m_i due to the same reason. Thus no other vehicle knows how to compose SK_{i1}. SK_{i2}, on the other hand, is defined as $sH(ID_{i2})$. Recall $ID_{i2} = VPK_i \oplus H(m_iID_{i1})$. Again, since no other vehicle knows m_i, only V_i can compute SK_{i2}. Therefore, no other vehicle can forge a valid signature by vehicle V_i. Note also that RSUs do not know the master secret s, and thus cannot forge a message either.

For local messages, the signature ς_i on message M_i by vehicle V_i is composed of LSK_{i1} and LSK_{i2}. LSK_{i1} is defined as sm_iLID_{i1}. Due to the difficulty of solving the discrete logarithm problem, there is no way for attackers to reveal m_i. LSK_{i2}, on the other hand, is defined as $sm_iH(LID_{i2})$. Again, since no other vehicle knows m_i, only V_i can compute LSK_{i2}. Therefore, no other vehicle can forge a valid signature by vehicle V_i. Again note that RSUs do not know the master secret s, and thus cannot forge a message either.

In practice, RSUs can be cracked easily and this is unavoidable. However, we can implement additional measures in our schemes to reduce the impact. For example, we can classify messages into different security levels. For critical messages, we can require them to be verified by TA instead of by RSUs. Or we can have another variation under which a message can only be trusted if it is verified by multiple consecutive RSUs. We believe with these measures, even if a few RSUs are cracked, the damage is limited.

Identity privacy preserving: We argue that if Decisional Diffie-Hellman (DDH) problem is hard, then the pseudo identity of a vehicle can preserve its real identity. The proof is as follows:

We consider a game between a challenger and an attacker. The challenger starts by giving a set of system parameters including P and P_{pub}. The attacker then freely chooses two verification public keys VPK_0 and VPK_1 and sends them to the challenger (these choices do not need to be random, the attacker can choose them in any way it desires). The challenger sets a bit $x = 0$ with probability $\frac{1}{2}$ and sets $x = 1$ with probability $\frac{1}{2}$. The challenger then sends the attacker the pseudo identity corresponding to VPK_b together with the group public key. The attacker tries to guess the value of x, and outputs its guess, x'. The attacker's advantage in the game is defined to be $Pr[x = x'] - \frac{1}{2}$. We say that our pseudo identity generation algorithm is semantically secure against a chosen plain text attack (CPA) if the attacker's advantage is negligible.

Next we assume that we have an algorithm A which runs in polynomial time and has a non-negligible advantage e as the attacker in the game described above. We will construct a DDH attacker B which has access to A and achieves a non-negligible advantage. B is given (P, aP, bP, T) as input. We let t denote the bit that B is trying to guess (i.e. $T = abP$ when $t = 0$ and is set randomly otherwise). B gives A $(P, P_{pub} = aP)$ as input. (Note that a now plays the role of s in our scheme.) A then chooses two verification public keys VPK_0 and VPK_1 which it has queried for the corresponding group public keys m_0P and m_1P before and sends them to B. B is playing the role of challenger here, so it sets a bit x randomly and generates the pseudo identity $ID = (ID_1, ID_2)$ where $ID_1 = raP$, $ID_2 = VPK_b \oplus H(rT)$ and r is a random nonce to send to A. B also sends A the group public key bP. (Note that b now plays the role of m_i in our scheme.) A sends B a bit x', which is its guess for x. B guesses that $t = 0$ if $x = x'$.

If $t = 0$ (so $T = abP$), then $ID_2 = VPK_b \oplus H(rabP) = VPK_b \oplus H(bID_1)$ is a valid pseudo identity. In this case, A will guess b correctly with probability $\frac{1}{2} + \epsilon$. Thus, $Pr[B\text{succeeds}|t =$

$0] = \frac{1}{2} + \epsilon$. If $t = 1$, we claim that $Pr[B\text{succeeds}|t = 1] = \frac{1}{2}$. To see why, we observe that when T is randomly chosen, $H(rT)$ cannot be cancelled by ID_1 and so there is no way to obtain VPK_b. Thus it reveals no information about x. In this sense, the value of x is hidden to A, so the probability that A will guess it is simply $\frac{1}{2}$. Hence, $Pr[B\text{succeeds}] = \frac{1}{2}(\frac{1}{2} + \epsilon) + \frac{1}{2}\frac{1}{2} = \frac{1}{2} + \epsilon/2$. Since ϵ is non-negligible, this shows that B violates the assumption that DDH is hard.
Furthermore, the random nonce r makes the pseudo identity of a vehicle different in different messages. Also since the verification public key VPK_i of a certain vehicle is different as seen by different RSUs, even if all RSUs collude, they have no way to trace a particular vehicle's travelling route.
Traceability: Section 5.4 shows that TA is able to trace a vehicle's real identity, thus traceability is satisfied.
Confidentiality: For one-to-one communications in a group, the message M_j to V_j is masked by the component $sxGPK_j$. To remove the mask, one has to present the first point xP, s and m_j. However, m_j is the shared secret between V_j and RSU. Also s is the master key of TA known by vehicles and TA only. Therefore, other than V_j, any other vehicles as well as RSU do not know how to obtain M_j.
For one-to-many communications in a group, any message is encrypted using the common group secret $\beta = s \times rr$. Since the partial secret rr is transmitted securely from RSU to each vehicle in the group, vehicles outside the group have no knowledge about it. In addition, since s is known by vehicles and TA only, RSU does not know how to decrypt the message either.

7.2 Analysis on bloom filter approach

This sub-section analyses our newly-proposed bloom filter approach in the verification notification phase. We first show that the probability of having false positives is very small if we set the parameters for the bloom filters appropriately, then we show that our message overhead is about 10 times lower than that under the RAISE protocol. Note that the IBV protocol does not have a notification phase, so we only compare ours with the RAISE protocol.
The probability of having a false positive in our bloom filter approach (i.e., Case 3 in Table 2) is equal to the probability that all k bits are set in one bloom filter while not all k bits are set in another bloom filter. Thus the probability of Case 3 is $Pr(Case3) = 2(1 - (1 - \frac{1}{m})^{kn})^k(1 - (1 - (1 - \frac{1}{m})^{kn})^k) \sim 2(1 - e^{-\frac{kn}{m}})^k(1 - (1 - e^{-\frac{kn}{m}})^k)$. Interestingly we find that the value of k that minimizes the false positive probability of a single bloom filter (i.e. $k = \frac{m \ln 2}{n}$) also minimizes Pr(Case 3) approximately (up to 5 decimal places) based on our empirical results. Hence we set the number of hash functions to $\frac{m \ln 2}{n}$ in our schemes and $Pr(Case3) \sim 2(0.6185^{\frac{m}{n}}(1 - 0.6185^{\frac{m}{n}}))$. It can be shown that when $\frac{m}{n} = 5$, $Pr(Case3)$ is about 0.16. When $\frac{m}{n} = 10$, $Pr(Case3)$ drops to 0.016 only. That is, if there are 100 signatures in a batch, on average only 1 to 2 signatures are affected by bloom filter false positive and need to be re-confirmed.
Now, we analyze the message overhead. Assume that there are n signatures in a batch. For the RAISE protocol, the HMAC() value sent by each vehicle is 16 bytes long while the H() value sent by the RSU in the notification phase is 16 bytes long per message. After that RSU signs the notification message using an ECDSA signature which is 56 bytes long. Together with a message header of 2 bytes, the total message overhead for verifying a batch of n signatures is $16n + 16n + 56 + 2 = 32n + 58$ bytes.
For our schemes, the ECC signature sent by each vehicle is 21 bytes long. In the notification phase, we use two bloom filters. To lower the false positive rate in any bloom filter, the total number of bits used in each bloom filter is set to 10 times the number of signatures in the batch

(i.e. $\frac{m}{n} = 10$). We have two bloom filters and so a total of $\frac{20n}{8} = 2.5n$ bytes are needed. We also use 2 bytes to represent the number of signatures in a batch. Together with a message header of 2 bytes, the total message overhead for verifying a batch of n signatures is $21n + 2.5n + 2 + 56 + 2 = 23.5n + 60$ bytes.

Note that when Case 3 occurs, additional message overhead is required for the re-confirmation procedures. If Case 3 only occurs in the first trial and does not occur in the second trial, the total message overhead for verifying a batch of n signatures becomes $23.5n + 60 + P(23.5n + 60) = (1 + P)(23.5n + 60)$ bytes where $P = Pr(Case3)$. Hence, if Case 3 occurs in all the first K trials and we switch to the hash approach after that, the total message overhead becomes $\sum_{i=1}^{k} P^i(23.5n + 60) + P^k(37n + 58)$ bytes. The component $P^k(37n + 58)$ represents the message overhead used for the hash approach after K trials. That is, 21 bytes for each ECC signature, 16 bytes for each H() value, 56 bytes for ECDSA signature and 2 bytes for message header. Since P is about 0.016, even if K is only 2, the overhead of our scheme is much lower than that of RAISE. And we found that as long as $K > 1$, the overhead is similar in different values of K since the probability of Case 3 is very low, so re-confirmation is quite unlikely.

In Fig. 5, we set the value of K to 1, 2, 3 and 5, respectively, and with $\frac{m}{n}$ set to 5 and 10. We can see that with all values of K and $\frac{m}{n}$, the message overhead by our schemes is far lower than that by the RAISE protocol due to the use of ECC signatures and bloom filters in the notification phase. For our schemes, when $\frac{m}{n} = 5$, the more the number of trials before switching to hash approach, the lower the message overhead. When $\frac{m}{n} = 10$, the lines with $K = 2$, $K = 3$ and $K = 5$ overlap. It means that with $\frac{m}{n} = 10$ and as long as $K > 1$, the probability of Case 3 is very low and so re-confirmation is quite unlikely.

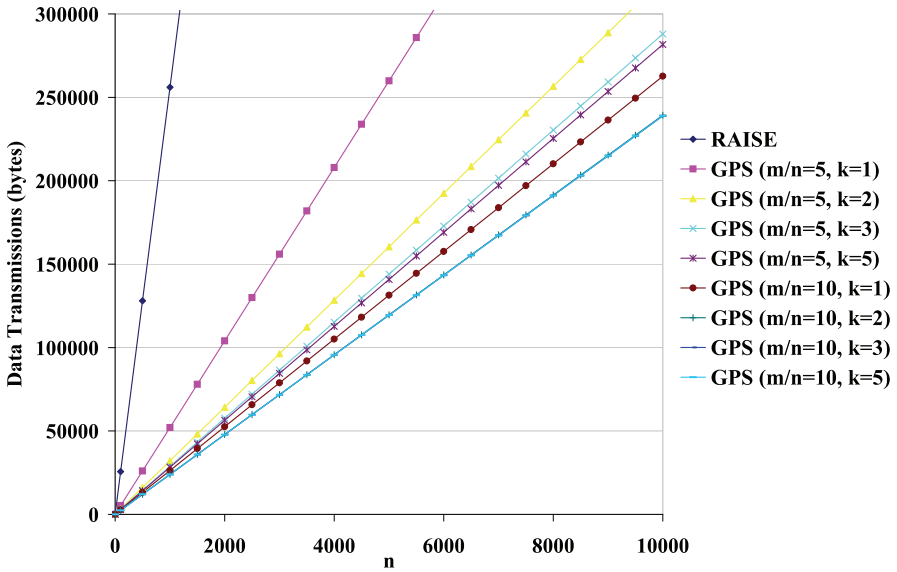

Fig. 5. Data transmission vs. number of signatures in the batch

8. Simulation results

In this section, we evaluate the network performance of our Grouping-enabled and Privacy-enhancing communications Scheme (GPS) in details. For ad hoc communications, we compare our scheme with the IBV protocol in terms of (1) the verification delay and (2) verification success rate through simulations. Note that IBV also uses a batch verification scheme, and so is much faster than the RAISE protocol. Thus, we compare the delay of our scheme with the IBV [Zhang, Lu, Lin, Ho & Shen (2008)] protocol. For success rate, we expect we will have a similar performance as RAISE as we will both identify all valid signatures even if there are invalid ones within the same batch. So, we compare our performance with the IBV protocol. We show that our scheme can verify more signatures while the additional delay required is insignificant. For group communications, we first compare our scheme with the RAISE protocol in terms of group message transmission delay. We expect we will have lower delay since group messages do not need to be verified by RSUs in our scheme. Next, we compare our GPS scheme with the SPECS protocol (group communications extension) [Chim, Yiu, Hui, Jiang & Li (2009)] in terms of (1) the group request delay and (2) group request success rate. We show that the success rate of forming a group using our scheme with the acknowledgement message is a lot higher than that of the SPECS scheme (with group communications extension) if the wireless channel is noisy. Then, we show that the delay introduced by retransmissions in our scheme is only marginal. Finally, we show the performance of our scheme in terms of key update average delay and member joining average delay as the number of vehicles in a group varies.

8.1 Simulation models

Some of the settings of our simulation are adopted from [Zhang, Lu, Lin, Ho & Shen (2008); Zhang, Lin, Lu & Ho (2008); Chim, Yiu, Hui, Jiang & Li (2009)]. We consider a highway of length 10 km and a number of RSUs are installed along it. The RVC and the IVC ranges are set to 600m and 300m, respectively. The bandwidth of the channel is 6 Mb/s and the average length of inter-vehicle message is 200 bytes. We compute the transmission time based on the bandwidth and the length of the message. The RSU performs batch verification every 300 ms and each pairing operation takes 4.5 ms [Scott (2007)]. We implement the simulation using C++.

For ad hoc communications, we assume that vehicles pass through an RSU at speeds varying from 50 km/h to 70 km/h. Our simulation runs for 1000 s. Inter-vehicle messages are sent every 500 ms from each vehicle. IEEE 802.11a is used to simulate the medium access control layer. (We simulate the IEEE 802.11a protocol by generating the time stamps for broadcasting messages of each vehicle. In case two stamps are identical, we randomly regenerate one of them.) We vary the total number of vehicles that have ever entered the RSU's RVC range during the simulation period from 200 to 1000 in steps of 200 to simulate the impact of different traffic densities. We also vary the inter-vehicle message signature error rate from 1% to 10% to study its impact on the performance of our scheme. For each configuration, we compute the average of 5 different random scenarios.

For group communications, the number of RSUs is a variable and these RSUs are evenly distributed along the given highway. Groups of vehicles are travelling on it at speeds varying from 50 km/h to 70 km/h. For each group, the number of vehicles is a variable and the vehicles are travelling on the road one after another. To simulate a noisy wireless channel (e.g. signal interference and signal blocking by obstacles or other signals), we use corruption rate. Since channel collision is also a kind of noise, we incorporate it into our

corruption rate as well for simplicity. If the corruption rate is $cr\%$, a transmitted message is corrupted with probability $cr\%$. We vary the corruption rate from 5% to 50% in steps of 5% to investigate its impact on the group request success rate and group request average delay. For each configuration, we compute the average of 20 different random scenarios (since we find that the standard deviation of the results is larger than that in experiments for ad hoc communications). The simulation runs for 1 hour and for every minute, there will be a new group of vehicles. For each group, one group request is issued.

8.2 Simulation results for ad hoc communications

We fix the signature error rate (the percentage of invalid signatures) to 5% and vary the total number of vehicles that have entered the RSU's range throughout the simulation. We only consider batches that contain invalid signatures (Invalid batch). In [Zhang, Lin, Lu & Ho (2008)], the expression for verification success rate is defined. We extend its definition to handle invalid batch: $IBSR = \frac{1}{N}\sum_{i=1}^{N}\frac{M_{app}^{i}}{M_{mac}^{i}}$, where M_{app}^{i} is the total number of messages that are successfully verified by the RSU and are consumed by vehicle V_i in the application layer before vehicle V_i leaves RSU's RVC range, M_{mac}^{i} is the total number of messages received by both vehicle V_i and RSU in the medium access control layer from other vehicles. For our scheme, we can have different levels of binary search as mentioned in Section 5. We use the notation GPS(BSx) to denote our scheme with x levels of binary search.

The verification success rates for the scheme are shown in Fig. 6. Note that the success rates for IBV and GPS(BS0) are 0% as both will drop the whole invalid batch. From our simulation, we found that even if we only have 1 level of binary search, the success rate of GPS(BS1) is already raised to about 45%. If we increase the number of levels to 4, the success rate can be raised to more than 90%.

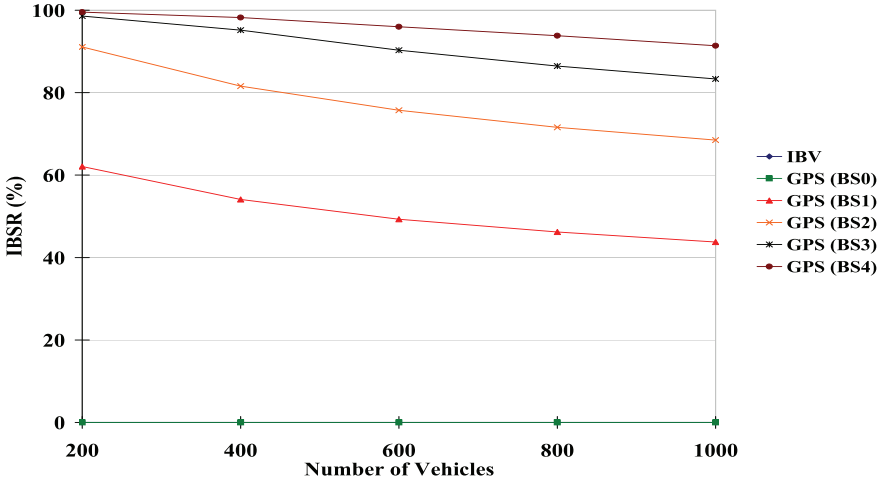

Fig. 6. Invalid batch success rate vs. number of vehicles

While the results are quite obvious, next we will show that the delay incurred by binary search procedure is minimal. Fig. 7 shows the delay performance. We define the average delay suffered by vehicles as $MD = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{M}\sum_{m=1}^{M}(T_{verf}^{m} - T_{recv}^{m})$, where M is the number of messages received by vehicle V_i, T_{verf}^{m} is the time that vehicle V_i receives the verification

notification message of message m from RSU and T^m_{recv} is the time that vehicle V_i receives message m from its neighboring vehicle. From Fig. 7, we can see that the delay under the IBV protocol and our scheme are very close to each other. For our scheme, as expected, with higher levels of binary search, longer delay is induced because more pairing operations are involved. However, even in the worst case (i.e. using 4 levels of binary search), our scheme only consumes an additional 10 ms which is roughly equivalent to the delay caused by 2 pairing operations. This is due to two main reasons. Not all cases require 4 levels of binary search and the time for each pairing operation is comparatively smaller than the transmission delay, so we can afford to do more pairing operations. One more interesting point to note is that without binary search, our scheme consumes 5 less ms than the IBV protocol. The reason is that our scheme requires 2 pairing operations only while the IBV protocol requires 3 as mentioned in Section 5.

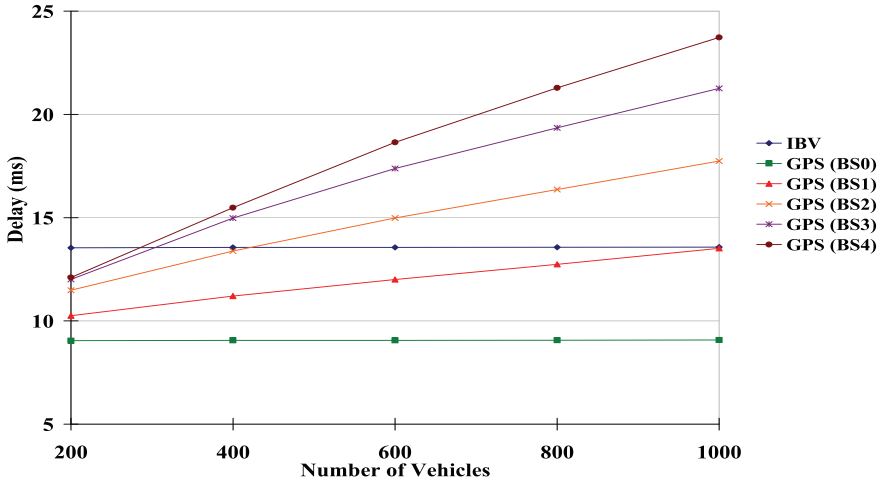

Fig. 7. Delay vs. number of vehicles

In the second set of experiments, we fix the number of vehicles that have entered RSU's RVC range during the simulation period to 300 and vary the signature error rate from 1% to 10% to investigate its impact on the invalid batch success rate and the message delay. We only consider batches that contain invalid signatures. Fig. 8 shows the results. The IBV and GPS(BS0) cases are not interesting as they drop all invalid batches. And it is also quite obvious that as the level of binary search increases, the success rate increases. The interesting point is that as the error rate increases from 1% to 10%, our scheme only degrades less than 10%.
The corresponding delay performance is shown in Fig. 9. As discussed earlier, GPS(BS0) gives a lower delay than the IBV protocol due to the saving of one pairing operation. As the error rate increases, more batches contain invalid signatures. Additional pairing operations are required to locate valid signatures. This increases the average delay. But the gap between our scheme and the IBV protocol is only about 10ms even when the error rate is 10%.

8.3 Simulation results for group communications

In the first set of experiments, we put aside the impact of interference and obstacles by setting the corruption rate to 0%. We vary the number of RSUs along the highway from 2 to 10. These RSUs are then evenly distributed along the highway. We define the group message

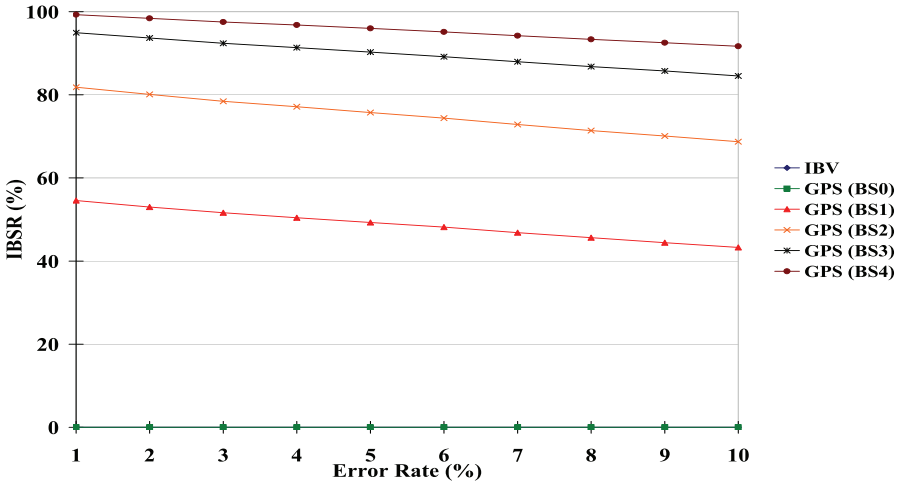

Fig. 8. Invalid batch success rate vs. error rate

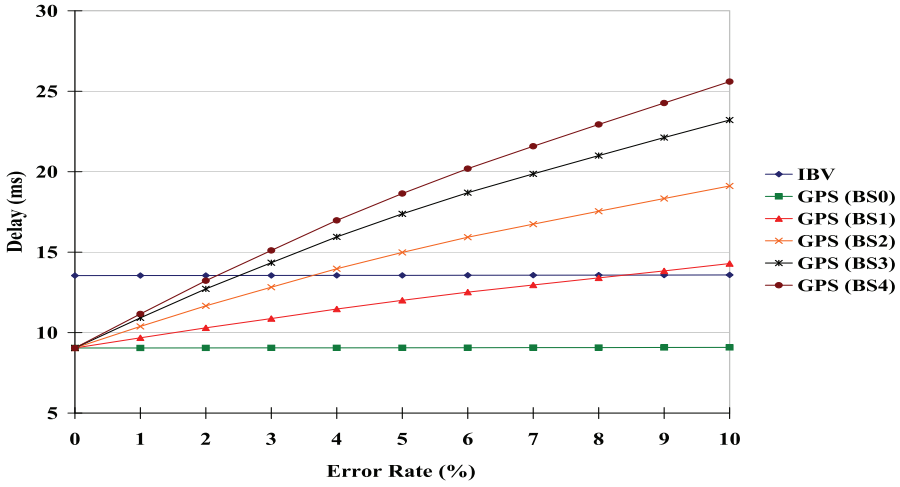

Fig. 9. Delay vs. error rate

transmission delay as the period from when a vehicle sends a message to when another vehicle in the same group verifies the message properly. We investigate the average group transmission delay under the RAISE protocol and our scheme as more RSUs are installed along the highway. From Fig. 10, we can see that under the RAISE protocol, the average group message transmission delay increases as RSUs become less dense along the highway. However, under our scheme, the average group message transmission delay remains constant (and near zero) no matter how dense the RSUs are. It is because under the RAISE protocol, all messages need to be verified by RSUs. When a vehicle wants to send a message to another group member, it first waits for a nearby RSU ahead of its journey. Then it waits for its batch verification period to expire. However, in our scheme, an initial RSU has already passed the necessary verification information (group public keys) to all vehicles in the group and so they know how to verify the signatures of each other without further support from RSUs. Thus the above two waiting periods are no longer needed.

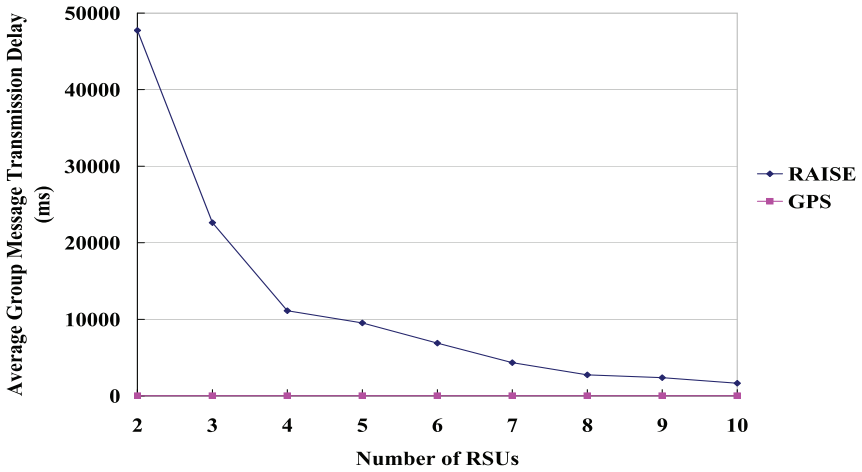

Fig. 10. Average group message transmission delay vs. number of RSUs

In the second set of experiments, we vary the corruption rate from 5% (low interference environment) to 50% (high interference environment) and study its impact on the group request success rate and the corresponding average delay. For each corruption rate, we further try different initial (before any new member joins) group sizes (n = 5, 10, 15 and 20). A group request is considered to be successful if all members in the group receive necessary information (i.e. group public keys of all others and the group partial secret) from the initial RSU for group communications. The group request success rate is defined as the number of successful requests divided by the total number of group requests made. The group request delay is defined a bit differently under SPECS and our GPS scheme. For the SPECS protocol, it is defined as the period from when any vehicle in the group first sends out a group request message to when all vehicles in the group receive necessary information for group communications. For our scheme, it is defined as the period from when any vehicle in the group first sends out a group request message to when RSU received and verified the acknowledgement messages from all vehicles in the group. The group request average delay is just the average of the group request delay among all successful group requests.
From Fig. 11, we can see that by requiring vehicles to acknowledge RSU, our scheme always

gives 100% group request success rate no matter how many vehicles are in the group. For SPECS, without any acknowledgement mechanism, the group request success rate drops gradually as the corruption rate increases from 5% to 50%. In particular, the more vehicles in the group, the lower the group request success rate. It is because with more vehicles in a group, the probability that all vehicles receive a message properly becomes lower.

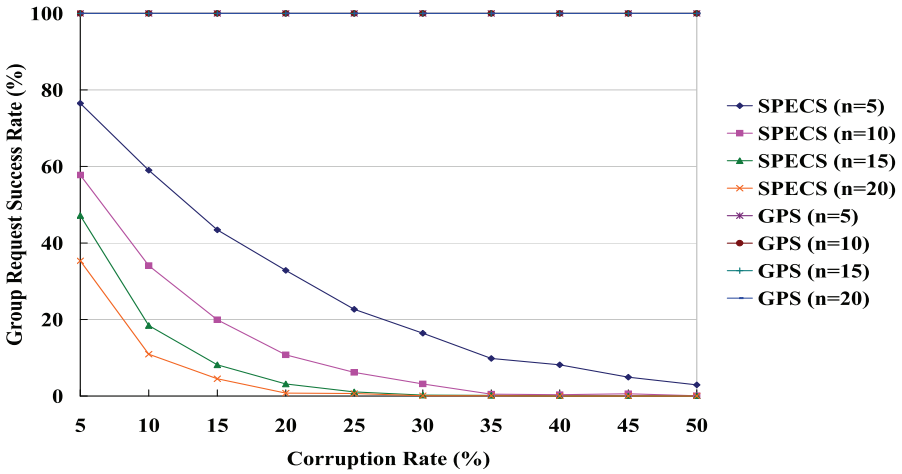

Fig. 11. Group request success rate vs. corruption rate

Fig. 12 shows the corresponding delay performance. We can see that larger groups are subjected to higher group request delay. This makes sense since with more vehicles in a group, the probability that all vehicles receive a message properly becomes lower. As a result, to ensure that all vehicles receive the message, more re-transmissions are needed. Thus higher delay is caused.

Next we focus on the case with 10 vehicles in a group to investigate the performance difference between SPECS and our scheme. From Fig. 13, we can see that our scheme gives a bit higher delay as the corruption rate increases due to increased number of re-transmissions. However, the increase is just marginal (less than 2 ms).

In the third set of experiments, we again vary the corruption rate from 5% to 50% and study its impact on the key update average delay under our scheme. A key update is considered to be successful if all vehicles in the group obtain the new group common secret. The key update delay is defined as the period from when any vehicle in the group sends out a key update request message to when it receives and verifies acknowledgement messages from all other vehicles in the group. The key update average delay is defined as the average of the key update delay among all successful key updates.

Fig. 14 shows that as more vehicles are involved in the group, higher key update delay is required. It is because with more vehicles in a group, the probability that all vehicles receive a message properly becomes lower. As a result, to ensure that all vehicles receive the message, more re-transmissions are needed. Thus higher delay is caused.

In the last set of experiments, we also vary the corruption rate from 5% to 50% but this time, we study its impact on the member joining average delay under our scheme. A member joining event is considered to be successful if all old members in the group obtain the new

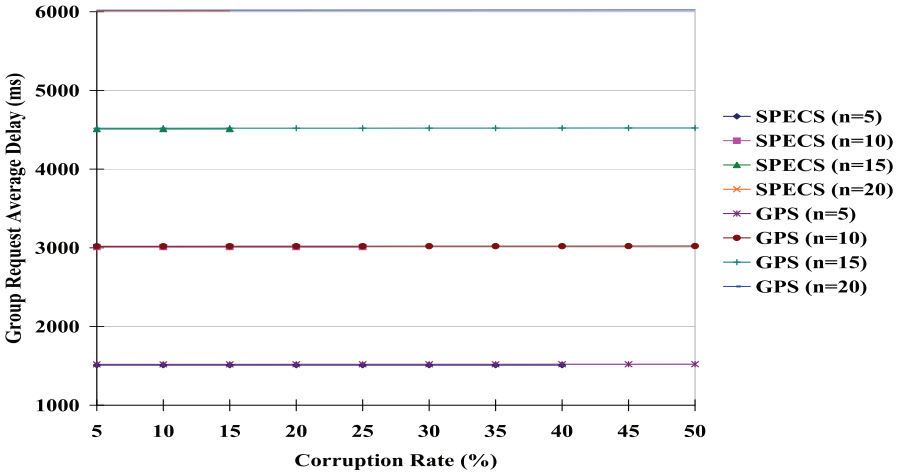

Fig. 12. Group request average delay vs. corruption rate

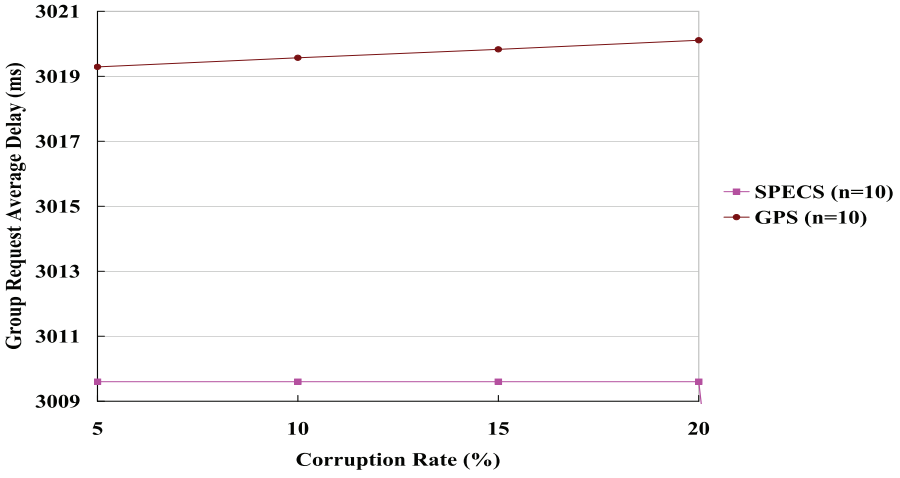

Fig. 13. Group request average delay vs. corruption rate

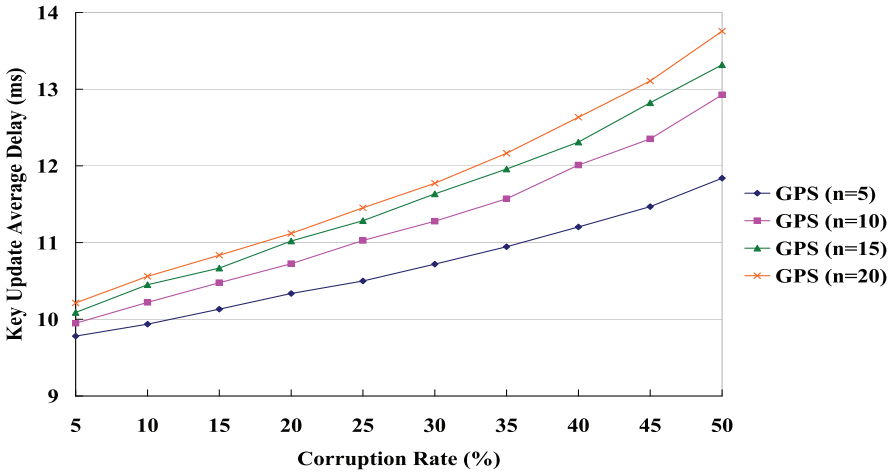

Fig. 14. Key update average delay vs. corruption rate

member's group public key while at the same time, the new member obtains all old members' group public keys and the group common secret for future communications. The member joining delay is defined as the period from when the new member sends out a group join request message to when the old member it contacts receives and verifies acknowledgement messages from all members. The member joining average delay is defined as the average of the member joining delay among all successful member joining events.

In Fig. 15, we study two initial (before any new member joins) group sizes ($n = 5$ and 10), and two new member set sizes, ($j = 5$ and 10) under our scheme. All configurations show slightly increasing trends as the corruption rate increases because of the same reason above. Also the delay performance under all configurations are actually close to each other and the difference is just about 10 ms.

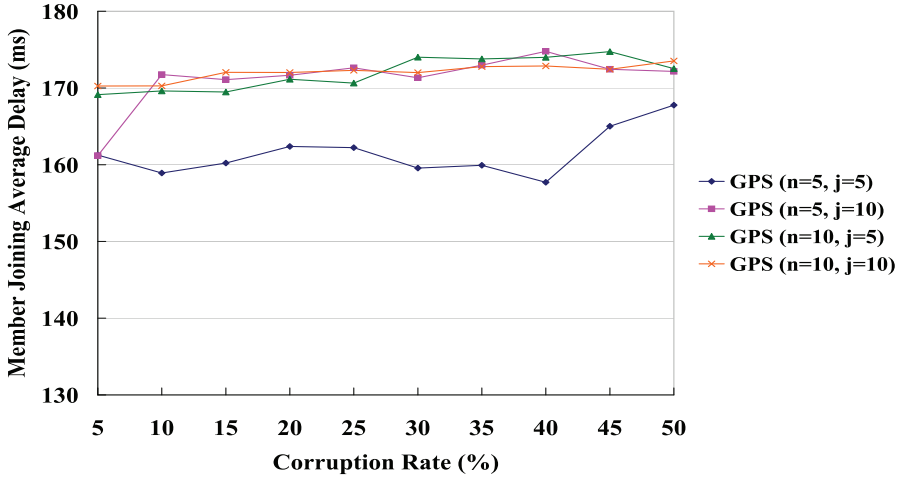

Fig. 15. Member joining average delay vs. corruption rate

9. Conclusions

In this chapter, we discussed the Grouping-enabled and Privacy-enhancing communications Schemes (GPS) for VANETs to handle ad hoc messages and group messages for inter-vehicle communications. For ad hoc messages, we follow the approach of letting RSU aid the signature verification process. We show that our schemes satisfy the security and privacy requirements. In terms of effectiveness, we show that our solution gives lower message overhead and at least 45 % higher success rate than previous work. For group messages, we proposed a set of modules for VANETs which allows dynamic membership and periodic update of the group key. RSU is needed only for group formation and member joining. In addition, we provide an add-on function for a group member to send a secure message to all other members or to a dedicated member. Again we show that our schemes satisfy all the security requirements. By simulation, we verify that our schemes are effective and the delay introduced by re-transmitting lost messages is negligible.
Note that in the early stage of VANET deployment, we may not have RSUs installed in all road sections. However, our protocol can be completed within the coverage of one RSU, and so can still be applied. Individual vehicles just cannot communicate on those sections of roads without RSUs, however, vehicles in the same group can still communicate without RSU. For future work, we will extend our group communications schemes to allow group merging and group splitting. We will also consider other secure applications in VANETs.

10. References

Boneh, D., Lynn, B. & Shacham, H. (2001). Short Signatures from the Weil Pairing, *Proceedings of Asiacrypt '01*, pp. 514 – 532.

Brown, R. H., Good, M. L. & Prabhakar, A. (1993). Data Encryption Standard (DES), *Federal Information Processing Standards (FIPS) Publication 46-2* .

Chim, T., Yiu, S., Hui, L. C., Jiang, Z. L. & Li, V. O. (2009). SPECS: Secure and Privacy Enhancing Communications Schemes for VANETs, *ADHOCNETS '09*.

Chim, T., Yiu, S., Hui, L. C. & Li, V. O. (2009). Security and Privacy Issues for Inter-vehicle Communications in VANETs, *IEEE Proceedings of the SECON '09 (Poster Session)*.

Eastlake, D. & Jones, P. (2001). US Secure Hash Algorithm 1 (SHA1), *IETF RFC3174* .

Housley, R., Ford, W., Polk, W. & Solo, D. (1999). Internet X.509 Public Key Infrastructure Certificate and CRL Profile, *IETF RFC2459* .

Hubaux, J. P., Capkun, S. & Lui, J. (2004). The Security and Privacy of Smart Vehicles, *IEEE Security and Privacy Magazine, 2(3)* pp. 49 – 55.

Kim, Y., Perrig, A. & Tsudik, G. (2004). Tree-based group key agreement, *ACM Transactions on Information Systems Security, 7(1)* pp. 60 – 96.

Menezes, A. (1991). An Introduction to Pairing-Based Cryptography, *1991 Mathematics Subject Classification, Primary 94A60*.

Oh, H., Yae, C., Ahn, D. & Cho, H. (1999). 5.8 GHz DSRC Packet Communication System for ITS Services, *IEEE Proceedings of the VTC '99*, pp. 2223 – 2227.

Raya, M., Papadimitratos, P. & Hubaux, J. P. (2006). Securing Vehicular Communications, *IEEE Wireless Communications Magazine, Special Issue on Inter-Vehicular Communications* pp. 8 – 15.

Scott, M. (2007). Efficient implementation of cryptographic pairings, Available online: http://ecrypt-ss07.rhul.ac.uk/Slides/Thursday/mscott-samos07.pdf.

Tsang, P. P. & Smith, S. W. (2008). PPAA: Peer-to-Peer Anonymous Authentication, *Proceedings*

of ACNS '08, pp. 55 – 74.
U.S. Department of Transportation, N. H. T. S. A. (2006). Vehicle Safety Communications Project Report.
Verma, M. & Huang, D. (2009). SeGCom: Secure Group Communication in VANETs, *IEEE Proceedings of the CCNC '09*, pp. 1 – 5.
Wang, F., Zeng, D. & Yang, L. (2006). Smart Cars on Smart Roads: an IEEE Intelligent Transportation Systems Society Update, *IEEE Pervasive Computing, Vol. 5, No. 4* pp. 68 – 69.
Wasef, A. & Shen, X. (2008). PPGCV: Privacy Preserving Group Communications Protocol for Vehicular Ad Hoc Networks, *IEEE Proceedings of the ICC '08*, pp. 1458 – 1463.
Wong, C. K., Gouda, M. & Lam, S. S. (1998). Secure group communications using key graphs, *IEEE Proceedings of the SIGCOMM '98*, pp. 68 – 79.
Zhang, C., Lin, X., Lu, R. & Ho, P. H. (2008). RAISE: An Efficient RSU-aided Message Authentication Scheme in Vehicular Communication Networks, *IEEE Proceedings of the ICC '08*, pp. 1451 – 1457.
Zhang, C., Lu, R., Lin, X., Ho, P. H. & Shen, X. (2008). An Efficient Identity-based Batch Verification Scheme for Vehicular Sensor Networks, *IEEE Proceedings of the INFOCOM '08*, pp. 816 – 824.

11. Appendix - attacks to IBV protocol

In this section, we first describe the IBV protocol. Then, we describe in details three security problems of the protocol - privacy violation, anti-traceability attack, and impersonation attack.

11.1 The IBV protocol

Before network deployment, TA sets up the parameters using the following steps:

1. TA chooses $\mathbb{G}$ and $\mathbb{G}_{\mathbb{T}}$ that satisfy the bilinear map properties.
2. TA randomly picks s_1, $s_2 \in \mathbb{Z}_q$ as its master keys. These two master keys are preloaded into each vehicle's tamper-proof hardware device.
3. TA then computes $P_{pub_1} = s_1 P$ and $P_{pub_2} = s_2 P$ as its public keys. The parameters $\{\mathbb{G}, \mathbb{G}_{\mathbb{T}}, q, P, P_{pub_1}, P_{pub_2}\}$ are then preloaded into all RSUs and OBUs.
4. TA also assigns each vehicle V_i a real identity $RID_i \in \mathbb{G}$ and a password PWD_i. The drivers are informed about them during network deployment or during vehicle first registration.

When a vehicle V_i starts up, its RID_i and PWD_i are input by the driver into a tamper-proof device. If they are valid, the tamper-proof device starts its role in generating pseudo identities, secret keys and message signing. Vehicle V_i's pseudo identity is generated as $ID_i = (ID_{i1}, ID_{i2})$ where $ID_{i1} = rP$ and $ID_{i2} = RID_i \oplus H(rP_{pub1})$ where r is a per-session random nonce. Its secret key is then generated as $SK_i = (SK_{i1}, SK_{i2})$ where $SK_{i1} = s_1 ID_{i1}$ and $SK_{i2} = s_2 H(ID_{i1}||ID_{i2})$. Here $H(.)$ is a MapToPoint hash function as in our schemes. When vehicle V_i wants to send the message M_i, it generates the signature $\sigma_i = SK_{i1} + h(M_i)SK_{i2}$ where $h(.)$ is a one-way hash function such as SHA-1. V_i then broadcasts ID_i, M_i and σ_i to the RSU.

RSU verifies the signature σ_i by checking whether $\hat{e}(\sigma_i, P) = \hat{e}(ID_{i1}, P_{pub1})\hat{e}(h(M_i)H(ID_{i1}||ID_{i2}), P_{pub2})$ as

$\hat{e}(\sigma_i, P)$

$= \hat{e}(SK_{i1} + h(M_i)SK_{i2}, P)$

$= \hat{e}(SK_{i1}, P)\hat{e}(h(M_i)SK_{i2}, P)$
$= \hat{e}(s_1 ID_{i1}, P)\hat{e}(h(M_i)s_2 H(ID_{i1}||ID_{i2}), P)$
$= \hat{e}(ID_{i1}, s_1 P)\hat{e}(h(M_i)H(ID_{i1}||ID_{i2}), s_2 P)$
$= \hat{e}(ID_{i1}, P_{pub1})\hat{e}(h(M_i)H(ID_{i1}||ID_{i2}), P_{pub2})$
Having the pseudo identity ID_i of vehicle V_i, TA can trace its real identity by using the *TA RID Tracing Routine*: $ID_{i2} \oplus H(s_1 ID_{i1}) = RID_i \oplus H(rP_{pub_1}) \oplus H(s_1 rP) = RID_i$.

11.2 Privacy violation

Any vehicle can obtain $ID_i = (ID_{i1}, ID_{i2})$ from V_i's transmissions. Also s_1 is preloaded into each vehicle's tamper-proof device during network deployment. Thus any vehicle can obtain V_i's RID_i by following the *TA RID Tracing Routine*.

11.3 Anti-traceability attack

We describe how a vehicle can make TA unable to trace its real identity from its message sent under the IBV protocol. We denote this kind of attack as an anti-traceability attack.
Assume that in a certain session, the attacking vehicle V_a generates its pseudo identity as $ID_a = (ID_{a1}, ID_{a2})$ where $ID_{a1} = rP$ and $ID_{a2} = GARBAGE \oplus H(aP_{pub1})$ where $GARBAGE \in \mathbb{G}$ and r is again a per-session random nonce. V_a then proceeds to generate its secret keys $SK_a = (SK_{a1}, SK_{a2})$ where $SK_{a1} = s_1 ID_{a1}$ and $SK_{a2} = s_2 H(ID_{a1}||ID_{a2})$, signs the message M_a by generating the signature $\sigma_a = SK_{a1} + h(M_a)SK_{a2}$ and sends out ID_a, M_a and σ_a to the RSU. Note that RSU can verify the message successfully because $\hat{e}(\sigma_a, P) = \hat{e}(ID_{a1}, P_{pub1})\hat{e}(h(M_a)H(ID_{a1}||ID_{a2}), P_{pub2})$. Assume that at a later time, V_a's message M_a causes an accident on the road. RSU forwards V_a's pseudo identity to TA so as to reveal V_a's real identity. However, upon computing $ID_{a2} \oplus H(s_1 ID_{a1}) = GARBAGE \oplus H(rP_{pub_1}) \oplus H(s_1 rP) = GARBAGE$, TA finds that $GARBAGE$ does not match any record at TA. V_a can thus evade its responsibility of causing the accident.

11.4 Impersonation attack

We describe how a vehicle can send messages on behalf of another under the IBV protocol. We denote this kind of attack as an impersonation attack.
Assume that at a certain instance, vehicle V_i with real identity RID_i generates its pseudo identity $ID_i = (ID_{i1}, ID_{i2})$, secret keys SK_i and signs message M_i by generating the signature σ_i as usual. While V_i is transmitting, an attacker V_a records ID_i. Later, V_a generates the message M_a. It generates its pseudo identity as $ID_a = (ID_{a1}, ID_{a2}) = ID_i = (ID_{i1}, ID_{i2})$ and its secret keys as $SK_a = (SK_{a1}, SK_{a2})$ where $SK_{a1} = s_1 ID_{a1} = s_1 ID_{i1}$ and $SK_{a2} = s_2 H(ID_{a1}||ID_{a2}) = s_2 H(ID_{i1}||ID_{i2})$. It then signs the message M_a by generating the signature $\sigma_a = SK_{a1} + h(M_a)SK_{a2}$ and sends out ID_a, M_a and σ_a to the RSU.
Similar to the anti-traceability attack, upon receiving V_a's message, RSU can verify it successfully because $\hat{e}(\sigma_a, P) = \hat{e}(ID_{i1}, P_{pub1})\hat{e}(h(M_a)H(ID_{i1}||ID_{i2}), P_{pub2})$. Assume at a later time, V_a's message M_a causes an accident on the road. RSU forwards V_a's pseudo identity ID_a as shown in its message to TA so as to reveal its real identity. After computing $ID_{a2} \oplus H(s_1 ID_{a1}) = ID_{i2} \oplus H(s_1 ID_{i1}) = RID_i \oplus H(rP_{pub_1}) \oplus H(s_1 rP) = RID_i$, both RSU and TA think that M_a is being sent by V_i because V_i's instead of V_a's identity is traced. Thus V_a can evade and pass its responsibility of causing the accident to V_i.