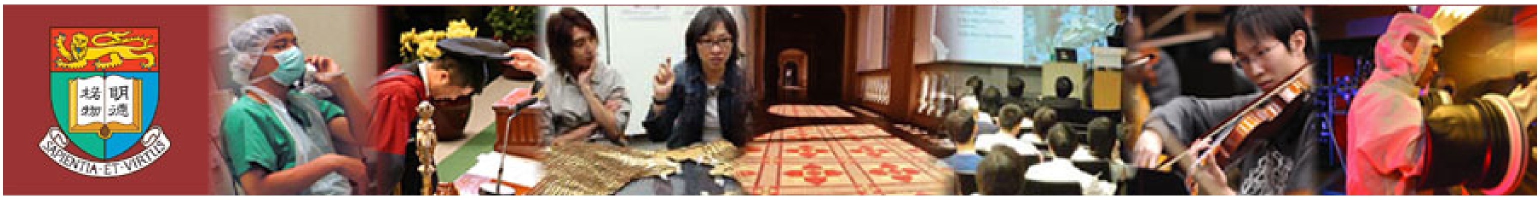

SPECS: Secure and Privacy Enhancing Communications Schemes for VANETs

T.W. Chim, S.M. Yiu and Lucas C.K. Hui
Department of Computer Science
The University of Hong Kong
Pokfulam Road, Hong Kong
Email: {twchim, smyiu, hui}@cs.hku.hk

Victor O.K. Li
Department of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam Road, Hong Kong
Email: vli@eee.hku.hk

***Abstract*—Vehicular ad hoc network (VANET) is an emerging type of networks which facilitates vehicles on roads to communicate for driving safety. The basic idea is to allow arbitrary vehicles to broadcast ad hoc messages (e.g. traffic accidents) to other vehicles. However, this raises the concern of security and privacy. Messages should be signed and verified before they are trusted while the real identity of vehicles should not be revealed, but traceable by authorized party. Existing solutions either rely heavily on a tamper-proof hardware device, or cannot satisfy the privacy requirement and do not have an effective message verification scheme. In this paper, we provide a software-based solution which makes use of only two shared secrets to satisfy the privacy requirement (with security analysis) and gives lower message overhead and at least 45% higher successful rate than previous solutions in the message verification phase using the bloom filter and the binary search techniques (through simulation study). We also provide the first group communication protocol to allow vehicles to authenticate and securely communicate with others in a group of known vehicles.**

***Index Terms*—Secure vehicular sensor network, authentication, batch verification, bloom filter, group communications**

I. INTRODUCTION

A vehicular ad hoc network (VANET) is also known as a vehicular sensor network 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) [1]. 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 [2] over the wireless channel while the RSUs, TA, and the application servers communicate using a secure fixed network (e.g. the Internet). The basic application of a VANET is to allow arbitrary vehicles to broadcast safety messages (e.g. road condition, traffic accident information) 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. This paper focuses on inter-vehicle communications.

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 before it can be relied on. 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 in recent years. 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 able to be revealed by a trusted party when necessary. For example, the driver who sent out fake messages causing an accident should not be able to escape by using an anonymous identity. Thus we call this kind of privacy conditional privacy.

In terms of integrity-checking and authentication, digital signature in conventional public key infrastructure (PKI) [3] is a well accepted choice. However, requiring a vehicle to verify the signatures of other vehicles by itself as in works like [4] induces two problems as mentioned in [5]. First, the computation power of an OBU is not strong enough 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 to help a vehicle to verify the message of another. The volume of signatures to be verified can be very huge (every vehicle is expected to broadcast a safety message every few hundred ms [6]). An efficient method for verifying a batch of signatures within a short period of time is desirable.

Related problems have been addressed in some recent works [5], [7]–[15]. In [7], 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 the Preliminaries Section 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]). Forth, 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, thus may imply a not satisfactory successful rate. Finally, the IBV protocol is not designed for vehicle-to-vehicle communications.

In a more recent work [5], 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 up to thousands of signatures within a short period of time, thus the notification messages induce a heavy message overhead.

Although the basic idea in an VANET is to allow unknown vehicles to broadcast safety message to one another, like other ad hoc network applications, there are scenarios (e.g. car racing, police patrolling, and tour travelling) which should allow a group of known vehicles to communicate securely among themselves. [8] considers such a secure group communications scenario but they only focus on how the group key can be updated. How vehicles can form a group and how the initial group key can be established are not considered at all.

Other recent efforts for making authentication in VANETs more efficient include [9] and [10]. In [9], the authors propose to use the physical property of a transmitting signal to discriminate one transmitter from others because physical measurement is more efficient than software computation. [10], on the other hand, aims at enhancing the efficiency of any certificate-based authentication scheme. The authors propose a HMAC-based solution to replace the time-consuming and traditional certificate revocation list checking process.

Regarding conditional privacy preserving, some recent works [11]–[13] propose to achieve the goal by using group signature schemes. That is, each vehicle in the system is assigned a group private key. When a vehicle wants to broadcast a message, it signs the message using its group private key. Verifiers such as RSUs can then verify its signature using a common group public key. In this way, a signature can be properly verified but at the same time, the real identity of the signer can be hidden. Only if necessary, a trusted party can use a private key to reveal the real identity of the signer. Though conditional privacy preserving can be achieved, we argue that such group signature schemes are complicated and inefficient.

In terms of secure VANET applications, [14] and [15] are two representatives. [14] proposes a secure navigation scheme for locating parking lots in a car park while [15] proposes a secure and privacy preserving road toll calculation scheme under the principle of multi-party computation.

In this paper, we propose two Secure and Privacy Enhancing Communications Schemes for vehicular sensor networks (SPECS). Our schemes can handle ”ad hoc messages” (those sent out 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 novel features over earlier schemes:

1) Our schemes are software based and do not rely on any special hardware. Our schemes are also based on bilinear pairing as in [7]. The pairing operation is known to be computationally expensive. We reduce the number of such operations in the verification phase from three to two to enhance the efficiency (a save of 33.3% of processing delay).
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. We also show that impersonation attack is not feasible in our schemes.
3) We make use of the techniques of binary search in RSU message verification phase and bloom filter to replace hash values in notification messages to reduce the message overhead substantially and enhance the effectiveness of the verification phase. Bloom filter is a well-known technique. We show an interesting application of it by using two bloom filters with opposite meaning to substantially reduce the false positive rate by up to 89.6%.
4) Any vehicle can form a group with other vehicles after an initial handshaking phase with a nearby RSU and then can authenticate and communicate with one another securely without the intervention of RSU even after moving into the region of another RSU.

We provide a security analysis on our schemes and an anlysis on the effectiveness of using bloom filter to replace hash values in the notification messages. Through the analysis and extensive simulation, we show that our schemes can reduce the message overhead and increase the successful rate by at least 45% while the additional overhead is insignificant when compared to the existing solutions.

The remainder of this paper is organized as follows: the system model and the problem statement are described in Section II. Some preliminaries about bilinear maps and bloom filter are given in Section III. Our schemes are presented in Section IV. The analysis and evaluation of our schemes are given in Sections V, VI and VII. Finally, Section VIII concludes the paper.

II. Problem Statement

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 followings:

[1] Please refer to the Appendix for details of the attacks.

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) The 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.
4) There exists a conventional public key infrastructure (PKI) for initial handshaking. The public key of the TA PK_{TA} is known by *everyone*. The public key of vehicle V_i PK_{V_i} is known by the TA. Also any RSU R broadcasts its public key PK_R with hello messages periodically to vehicles that are travelling at the RVC range of it. Thus PK_R is known by all vehicles nearby. There is no need for vehicles to know the public keys of other vehicles to avoid message overhead for exchanging certificates. The private keys of TA, V_i and R are SK_{TA}, SK_{V_i} and SK_R respectively and are kept secret by the corresponding party.
5) The real identity of any vehicle is only known by the TA and itself but not by others.

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 and revocability: Although a vehicle's real identity should be hidden from other vehicles, if necessary, the TA should have the ability to obtain a vehicle's real identity and to revoke it from future usage.

III. Preliminaries

Our schemes are *pairing-based* and defined on two cyclic groups with a *bilinear mapping* [16]. We briefly introduce what a bilinear map is and will discuss the basics on bloom filter which we apply in the RSU notification phase.

A. Bilinear Maps

Let $\mathbb{G}$ be a cyclic additive group 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.

B. 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 $\frac{mln2}{n}$.

IV. Our Solutions - SPECS

This section presents our proposed SPECS schemes. 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 a real identity $RID \in \mathbb{G}$ and a password PWD. The drivers are informed about them during network deployment or during vehicle first registration.

The schemes can be divided into the following modules:

1) **Initial handshaking (Fig. 1)**: This module is executed when a vehicle meets a new RSU. The vehicle authenticates itself with the TA via RSU. Note that TA is the only authorized party to know 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 the 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 journey. To increase the security level, s is not preloaded into any hardware on the vehicle like [7]. For the shared secret with RSU, a new secret is generated every time the vehicle moves into the region of another RSU.

   For ad hoc messages, we have the following modules:

2) **Message signing (Fig. 2)**: 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 to trace its real identity. Then, it signs the message using the signing key of the pseudo identity.
3) **Batch verification (Fig. 3)**: 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 [7], once there is an invalid signature in the batch, the whole batch of signatures are assumed to be invalid and ignored).
4) **Real identity tracking and revocation**: This module is used by TA to reveal the real identity of the sender of a given message and then revoke it from future usage if necessary.

   For group messages, we have the following modules:

5) **Group key generation (Fig. 4)**: This module is used when a set of vehicles want to form a group. A group secret key will be generated by the TA and forwarded by an RSU.
6) **Group message signing and verification (Fig. 5)**: This module shows how to generate a group message so that the group members can verify the signature without the help of an RSU. Note that to reveal the real identity of the sender of a group message by the TA, we can apply the same procedure as for ad hoc message.

A. Initial handshaking

We use the notations $ENC_Z(M)$, $DEC_Z(M)$ and $SIG_Z(M)$ to denote encrypting, decrypting and signing, respectively, message M using the key Z from now on. The detailed processes in this module are as follows:

1) When a vehicle V_i meets the first RSU R, it signs its RID and PWD using its private key SK_{V_i}. It then encrypts RID, PWD and $SIG_{SK_{V_i}}(RID, PWD)$ using the TA's public key PK_{TA} and sends $ENC_{PK_{TA}}(RID, PWD, SIG_{SK_{V_i}}(RID, PWD))$ to the RSU which forwards it to the TA.

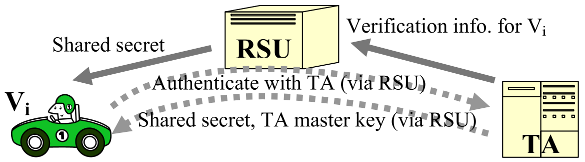
Shared secret
RSU
Verification info. for V$_i$
Authenticate with TA (via RSU)
V$_i$
Shared secret, TA master key (via RSU)
TA

Fig. 1. Initial Handshaking

2) The TA decrypts the block and verifies RID, PWD and checks V_i's signature using its public key PK_{V_i}. If they are all valid and if RID is not in its revocation list, it generates a shared secret t_i for V_i and computes V_i's ID Verification Public Key as $VPK_i = t_i \oplus RID$. TA then passes VPK_i to the RSU to enable it to verify signatures from V_i even if V_i uses pseudo identity to sign the message. The TA then stores the (RID, 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 the RSU, where PK_R and PK_{V_i} are conventional public keys of the RSU and vehicle V_i respectively. Note that to let V_i know that s and VPK_i are really sent by the TA, the TA includes its signature on s and VPK_i ($SIG_{SK_{TA}}(s, VPK_i)$) into the encrypted text.
3) The 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 $Y = ENC_{PK_{V_i}}(m_i, SIG_{SK_R}(m_i))$ and X to vehicle V_i. Again to let vehicle V_i know that m_i is really sent by the RSU, the RSU signs it.
4) Vehicle V_i decrypts Y to obtain m_i and verifies the RSU's signature on it. Similarly, it decrypts X to obtain s and VPK_i and verifies the TA's signature on them. It then computes its shared secret with the TA using $t = VPK_i \oplus RID$.

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 the 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 generate a random nonce r' and sends $ENC_{PK_{TA}}(RID||r')$ to TA via this new RSU. The random nonce r' avoids V_i from being tracked even the attacker captures a number of these packets as V_i is moving across RSUs. The TA obtains RID by decrypting the block using its private key SK_{TA} and then removing the concatenation r'. This time the 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. The 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 $Y = ENC_{PK_{V_i}}(m_i, SIG_{SK_R}(m_i))$ 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.

B. Message signing

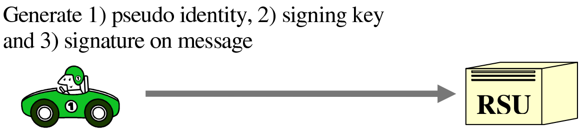
Generate 1) pseudo identity, 2) signing key and 3) signature on message
RSU

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 [17]. 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 [18]. Vehicle V_i then sends $< ID_i$, M_i, $\sigma_i >$ to others.

C. Batch verification

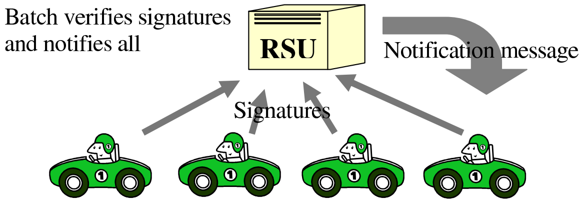

Fig. 3. Batch Verification

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 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 [7].

Verification procedure. Assume that the 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, the RSU first finds out 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 $\hat{e}(\sum_{i=1}^{n} \sigma_i, P) = \hat{e}(\sum_{i=1}^{n} m_i ID_{i1} + h(M_i)H(ID_{i2}), P_{pub})$.

Proof of correctness:
L.H.S.
$= \hat{e}(\sum_{i=1}^{n} SK_{i1} + h(M_i)SK_{i2}, P)$
$= \hat{e}(\sum_{i=1}^{n} SK_{i1}, P)\hat{e}(\sum_{i=1}^{n} h(M_i)SK_{i2}, P)$
$= \hat{e}(\sum_{i=1}^{n} sm_i ID_{i1}, P)\hat{e}(\sum_{i=1}^{n} h(M_i)sH(ID_{i2}), P)$
$= \hat{e}(\sum_{i=1}^{n} m_i ID_{i1}, sP)\hat{e}(\sum_{i=1}^{n} h(M_i)H(ID_{i2}), sP)$
$= \hat{e}(\sum_{i=1}^{n} m_i ID_{i1}, P_{pub})\hat{e}(\sum_{i=1}^{n} h(M_i)H(ID_{i2}), P_{pub})$
$=$ R.H.S. □

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, the RSU reject the message immediately. Note that if a vehicle does not know the shared secret with the 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 the 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 discussed in Section III-B, 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 I). 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 the RSU still has not yet verified σ_j and so V_i has to wait for the 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 the 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 the RSU. RSU searches for σ_j from those stored signatures. If σ_j can be found, the RSU adds the hash of V_j into the positive filter.

TABLE I
POSSIBLE CASES AND THEIR IMPLICATIONS IN BLOOM FILTERS

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)

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.

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, the RSU will send $h(ID_j||M_j)$ of V_j to vehicle V_i as a direct notification. To facilitate the RSU to know what it should send in the re-confirmation reply, the RSU stores the number of requests to each of its signature stored. See 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 interpret the size the filters (so that they can adjust the range of hash functions accordingly), together with the valid and the invalid filters, the 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 up to thousands of signatures depending on the traffic density around the 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 find out 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 VII, 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.

D. Real identity tracking and revocation

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 $ID_{i2} \oplus t_j \oplus H(m_i ID_{i1}) = RID_j$.

Proof of correctness:
L.H.S.
$= t_i \oplus RID_j \oplus H(m_i ID_{i1}) \oplus t_i \oplus H(m_i ID_{i1})$
= R.H.S. □

No other party can obtain vehicle V_i's real identity since t_i is only known by the TA and V_i itself.

Upon getting V_i's real identity RID_i, TA can revoke it if necessary. This can be done by simply storing RID_i into a revocation list. V_i can no longer obtain VPK_i from it in the future.

E. Group key generation

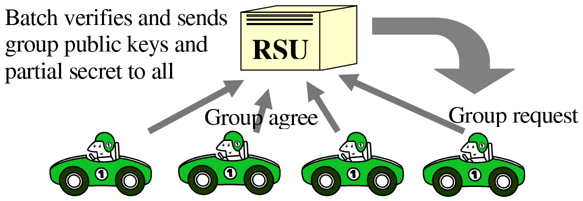
Batch verifies and sends group public keys and partial secret to all
RSU
Group agree
Group request

Fig. 4. Group Key Generation

This subsection shows how a group of known vehicles can form a group with any RSU, then they can communicate securely within the group without any further help from RSU to verify these group messages.

Assume that vehicles V_1, V_2, ..., V_n have already registered with an RSU and their shared secrets with the RSU are m_1, m_2, ..., m_n respectively. Also assume that these vehicles know pseudo identities of one another already or they can know others' pseudo identities by the last message received from one another.

Group request. Vehicle V_i first sends to the RSU message $M_i = \{GPREQ, ID_1, \ldots, ID_{i-1}, ID_{i+1}, ..., ID_n\}$ and its signature $\sigma_i = SK_{i1} + h(M_i)SK_{i2}$ on it where ID_j is the pseudo identity of V_j. Also SK_{i1} and SK_{i2} are generated using the methods in Section IV-B. 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 pseudo identity is included in the $GPREQ$ message. If yes, it sends out $M_j = \{GPAGR, ID_j\}$ and its signature $\sigma_j = SK_{j1} + h(M_j)SK_{j2}$.

Group batch verification. The RSU then batch-verifies σ_1, ..., σ_n. For any vehicle V_x whose signature is found to be valid, it generates its group public key as $GPK_x = m_x P$. Recall that m_x is the shared secret between the RSU and vehicle V_x. Besides group public keys, the RSU also requests the TA to provide the group of vehicles a common group secret key. Without loss of generality, assume the signatures from V_1, ..., V_x are valid. The RSU sends VPK_1, ..., VPK_x to the TA which in turn generates a random number rr and computes the group secret key as $CGS = s \times rr$. Next the TA sends $ENC_{t_1}(CGS)$, ..., $ENC_{t_x}(CGS)$ back to the RSU. Recall that t_i is the shared secret between V_i and the TA. The RSU then broadcasts $M_r = \{ID_1, ..., ID_x, GPK_1, ..., GPK_x, ENC_{t_1}(rr), ..., ENC_{t_x}(rr)\}$ and its signature $SIG_{SK_R}(M_r)$ to the vehicles concerned. Note that in case the verification fails due to invalid signatures or vehicles inside the range have 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 reason.

Group secret establishment. Each vehicle in the group stores all the group public keys and the decrypted CGS values. Note that the RSU does not know CGS since the TA encrypts it using its shared secret with each vehicle. Thus vehicles in the group can communicate with others securely from now on.

F. Group message signing and verification

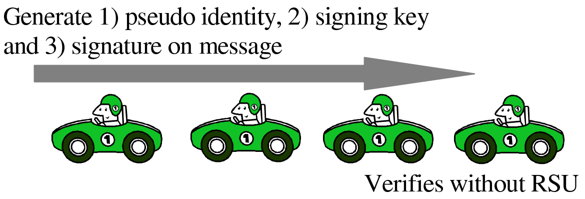
Generate 1) pseudo identity, 2) signing key and 3) signature on message
Verifies without RSU

Fig. 5. Group Message Signing and Verification

Next we look at the pseudo identity generation, message signing and signature verification when group communications take place. When vehicle V_i wants to send a group message M_i, it generates its pseudo identity ID_i and signature σ_i in the same way as in Section IV-B. However, its secret signing key is generated as $SK_i = (SK_{i1}, SK_{i2})$ where $SK_{i1} = m_i ID_{i1}$ and $SK_{i2} = m_i H(ID_{i2})$. V_i then sends out $< ID_i$, $ENC_{CGS}(GPK_i || ID_i)$, M_i, $\sigma_i >$ where r is the random nonce used to generate its pseudo identity. Note that GPK_i is included so that the receiving vehicle knows which group public key to use for verification. To make it impossible for any vehicle outside the group to trace V_i, GPK_i is first concatenated with its per session pseudo identity and then encrypted using the common group secret CGS.

To verify the signature σ_i of vehicle V_i on message M_i, the receiving vehicle first decrypts $ENC_{CGS}(GPK_i || ID_i)$ using CGS. If it finds that GPK_i obtained does not belong to any group member, it simply ignores the message. Otherwise it checks whether $\hat{e}(\sigma_i, P) = \hat{e}(ID_{i1} + h(M_i)H(ID_{i2}), GPK_i)$.

Proof of correctness:

L.H.S.
$= \hat{e}(SK_{i1} + h(M_i)SK_{i2}, P)$
$= \hat{e}(SK_{i1}, P)\hat{e}(h(M_i)SK_{i2}, P)$
$= \hat{e}(m_i ID_{i1}, P)\hat{e}(h(M_i)m_i H(ID_{i2}), P)$
$= \hat{e}(ID_{i1}, m_i P)\hat{e}(h(M_i)H(ID_{i2}), m_i P)$
$= \hat{e}(ID_{i1}, GPK_i)\hat{e}(h(M_i)H(ID_{i2}), GPK_i)$
$=$ R.H.S. □

V. Security Analysis

We analyse our schemes to show that they are secure with respect to the security requirements listed in section II. For the first two requirements - message integrity and authentication, identity privacy preserving, we try to prove the security formally using ways discussed in [19], [20] and [21]. Basically, we define some games which actually represent some possible ways that an attacker can attack our system. Then through some mathematical calculations, we show that all these games are unlikely and so our schemes are secure.

A. Message integrity and authentication

In this sub-section, we show that an attacker cannot generate a valid signature on behalf of any vehicle. We consider two different kinds of attackers: 1) an attacker is itself a vehicle and 2) an attacker is an RSU (e.g. an RSU being hacked).

a. Vehicle as an attacker:

We first consider the case that the attacker is a vehicle (that is he knows s but not m_i for any other vehicle V_i). We argue that if DH is hard, then a vehicle's message (either ad hoc message or group message) cannot be forged by another vehicle and our scheme is secure against existential forgery, adaptive chosen message attack under random oracle model. The proof is as follows.

We first consider **Game 1** between a challenger and an attacker who is a vehicle:

Setup: The challenger starts by giving the attacker a set of system parameters including P and P_{pub}. The challenger also gives the attacker a pseudo identity $ID_i = (ID_{i1}, ID_{i2})$ and the corresponding group public key GPK_i. This simulates the situation that the attacker eavesdropped ID_i and GPK_i of another vehicle V_i from the air. Further since s is known by all vehicles in our scheme, the challenger sends it to the attacker as well.

Query: Assume the attacker does not know how to compute $H(.)$ and $h(.)$ functions. He can ask the challenger for the value $H(ID_{i2})$ and the hash ($h(.)$) values of up to k different messages.

Challenge: The challenger then asks the attacker to pick two random message M_i and M_j and sign them on behalf of V_i to produce σ_i and σ_j.

Guess: Finally, the attacker sends two pairs $< M_i, \sigma_i >$ and $< M_j, \sigma_j >$ to the challenger.

The attacker's advantage in this game is defined to be $Pr[\sigma_i$ and σ_j are valid signatures]. We say that our signature scheme is secure against existential forgery, adaptive chosen message attack 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 **Game 1**. We will construct **Game 2** in which a Diffie-Hellman (DH) attacker B can make use of A to achieve a non-negligible advantage in breaking DH. B is given the values P, a, aP, bP, cP and d, where a, b, c and d are some constants, as inputs and he is asked to compute $ab(2c+d)P$. Note that computing $ab(2c+d)P$ is as hard as computing bcP. Now let us look at how B can make use of A to solve this DH problem by following the steps below:

Setup: B makes up the parameters $(P, P_{pub} = aP)$ in our SPECS scheme. Note that a now plays the role of s. Since s is known by all vehicles in our scheme, B sends a to A as well. Further B also provides A a pseudo identity $ID_i = (ID_{i1}, ID_{i2}) = (cP, xP)$, where x is a random number, and the corresponding group public key $GPK_i = bP$ where b plays the role of shared secret m_i in our scheme. Note that A, as a vehicle, has no way to validate the given pseudo identity since only TA and RSUs have such an ability. Thus A will not doubt the relationship between ID_{i1} and ID_{i2}.

Query: A then asks B for the value $H(ID_{i2})$ (this is the only $H(.)$ value it needs to impersonate V_i) and B replies with bP. Next A picks up to k random messages and queries B for their hash ($h(.)$) values. B answers these queries using a random oracle. B maintains a table to store all its answers. Upon receiving a message, if the message has been queried before, B answers with the stored value. Otherwise, it answers with a random value and stores it into its table for later usage. Except for the u^{th} and v^{th} queries (say messages M_u and M_v), B answers with the values x and $d - x$ respectively where $x < d$ is a random number.

Challenge: When the query phase is over, B asks A to pick two random messages M_i and M_j and sign them on behalf of V_i.

Guess: A picks two random messages M_i and M_j, generates signatures σ_i and σ_j on them on behalf of V_i and sends the pairs $< M_i, \sigma_i >$ and $< M_j, \sigma_j >$ to B. Note that A must have queried M_i and M_j in the query phase, otherwise he does not know how to compute $h(M_i)$ and $h(M_j)$.

If $M_i = M_u$ and $M_j = M_v$ or $M_i = M_v$ and $M_j = M_u$, B computes $\sigma_i + \sigma_j$. This is equivalent to $abID_{i1} + aH(ID_{i2})h(M_i) + abID_{i1} + aH(ID_{i2})h(M_j) = ab(cP) + a(bP)(x) + ab(cP) + a(bP)(d-x) = ab(2c+d)P$. Having this value, B resolves the given DH instance successfully. Assume A's advantage in breaking our SPECS scheme is ϵ and the probability that A picks M_u and M_v is $1/C(k,2)$. Hence, $Pr[B$ succeeds$] = 1/C(k,2) \times \epsilon$. Since ϵ is non-negligible, B can solve the DH problem but this violates the assumption that DH is hard. Therefore, our signature scheme is secure against existential forgery, adaptive chosen message attack under random oracle model. □

b. RSU as an attacker:

Next we consider the case that the attacker is an RSU (that is he knows m_i for some vehicle V_i but not s). We argue that if DH is hard, then a vehicle's message (either ad hoc message or group message) cannot be forged by an RSU and our scheme is secure against existential forgery, adaptive chosen message attack under random oracle model. The proof is as follows.

We first consider **Game 1** between a challenger and an attacker who is an RSU:

Setup: The challenger starts by giving the attacker a set of system parameters including P and P_{pub}. The challenger also gives the attacker a pseudo identity $ID_i = (ID_{i1}, ID_{i2})$, the corresponding group public key GPK_i and shared secret m_i. This simulates the situation that the attacker eavesdropped ID_i of any vehicle V_i from the air and that he is the RSU assigning GPK_i and m_i.

Query: Assume the attacker does not know how to compute $H(.)$ and $h(.)$ functions. He can ask the challenger for the value $H(ID_{i2})$ and the hash ($h(.)$) values of up to k different messages.

Challenge: The challenger then asks the attacker to pick two random messages M_i and M_j and sign them on behalf of V_i to produce σ_i and σ_j.

Guess: Finally, the attacker sends two pairs $< M_i, \sigma_i >$ and $< M_j, \sigma_j >$ to the challenger.

The attacker's advantage in this game is defined to be $Pr[\sigma_i$ and σ_j are valid signatures$]$. We say that our signature scheme is secure against existential forgery, adaptive chosen message attack 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 **Game 1**. We will construct **Game 2** in which a Diffie-Hellman (DH) attacker B can make use of A to achieve a non-negligible advantage in breaking DH. B is given the values P, aP, b, bP, cP and d, where a, b, c and d are some constants, as inputs and he is asked to compute $ab(2c+d)P$. Note that computing $ab(2c+d)P$ is as hard as computing acP. Now let us look at how B can make use of A to solve this DH problem by following the steps below:

Setup: B makes up the parameters $(P, P_{pub} = aP)$ in our SPECS scheme. Note that a now plays the role of s. B also provides A a random verification public key VPK_i, a pseudo identity $ID_i = (ID_{i1}, ID_{i2}) = (cP, VPK_i \oplus H(bID_{i1}))$, the corresponding group public key $GPK_i = bP$ and b which plays the role of shared secret m_i in our scheme. Note that A is an RSU and it knows how to validate the composition of a pseudo identity. Therefore ID_i must be properly formed so that A will not have any doubt on it.

Query: A then asks B for the value $H(ID_{i2})$ (this is the only $H(.)$ value it needs to impersonate V_i) and B replies with bP. Next A picks up to k random messages and queries B for their hash ($h(.)$) values. B answers these queries using a random oracle. B maintains a table to store all its answers. Upon receiving a message, if the message has been queried before, B answers with the stored value. Otherwise, it answers with a random value and stores it into its table for later usage. Except for the u^{th} and v^{th} queries (say messages M_u and

M_v), B answers with the values x and $d-x$ respectively where $x<d$ is a random number.

Challenge: When the query phase is over, B asks A to pick two random messages M_i and M_j and sign them on behalf of V_i.

Guess: A picks two random messages M_i and M_j, generates signatures σ_i and σ_j on them on behalf of V_i and sends σ_i and σ_j to B. Note that A must have queried M_i and M_j before, otherwise he does not know how to compute $h(M_i)$ and $h(M_j)$.

If $M_i = M_u$ and $M_j = M_v$ or $M_i = M_v$ and $M_j = M_u$, B computes $\sigma_i + \sigma_j$. This is equivalent to $abID_{i1} + aH(ID_{i2})h(M_i) + abID_{i1} + aH(ID_{i2})h(M_j) = ab(cP) + a(bP)(x) + ab(cP) + a(bP)(d-x) = ab(2c+d)P$. Having this value, B resolves the given DH instance successfully. Assume A's advantage in breaking our SPECS scheme is ϵ and the probability that A picks M_u and M_v is $1/C(k,2)$. Hence, $Pr[B \text{ succeeds}] = 1/C(k,2) \times \epsilon$. Since ϵ is non-negligible, B can solve the DH problem but this violates the assumption that DH is hard. Therefore, our signature scheme is secure against existential forgery, adaptive chosen message attack under random oracle model. □

Note that the signature schemes for ad hoc messages and group messages only differ in the composition of pseudo identities. However, in the above proof, V_i's pseudo identity is provided to the attacker and its composition does not affect his forging process. Therefore, the proof above applies to both ad hoc messages and group messages.

In practice, RSUs can be cracked easily and this is unavoidable. However, we can add in additional measures to our schemes to reduce the impact. For example, we can classify messages into different security levels. For critical message, 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 effect is not a disaster.

B. *Identity privacy preserving*

In this sub-section, we show that an attacker cannot obtain a vehicle's real identity easily. Since the only information that is related to a vehicle's real identity and is exposed in the network is its pseudo identity, we show that an attacker cannot obtain a vehicle's real identity even it is keeping its pseudo identity.

We argue that if DDH is hard, then the pseudo identity of a vehicle can preserve its real identity. The proof is as follows.

We first consider **Game 1** between a challenger and an attacker:

Setup: The challenger starts by giving the attacker a set of system parameters including P and P_{pub}.

Choose: 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).

Challenge: The challenger sets a bit $x=0$ with probability 1/2 and sets $x=1$ with probability 1/2. The challenger then sends the attacker the pseudo identity corresponding to VPK_b together with the group public key.

Guess: The attacker tries to guess the value of x chosen by the challenger, and outputs its guess, x'.

The attacker's advantage in this game is defined to be $Pr[x=x'] - 1/2$. 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 **Game 1**. We will construct **Game 2** in which a Decisional Diffie-Hellman (DDH) attacker B can make use of A to achieve a non-negligible advantage in breaking DDH. B is given a DDH instance (P, aP, bP, T) as input and he is asked to determine whether $T = abP$. We further let t denote a bit that B is trying to guess (i.e. $t=0$ for positive answer $T=abP$ while $t=1$ for negative answer $T \neq abP$). **Game 2** runs as follows:

Setup: Based on the DDH instance, B makes up the parameters $(P, P_{pub} = aP)$ and gives them to A. Note that a now plays the role of s in our SPECS scheme.

Choose: 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 respectively, before and sends them to B.

Challenge: 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 $ID_1 = raP$, $ID_2 = VPK_x \oplus H(rabP)$ and r is a random nonce and sends to A. B also sends A the group public key bP. (Note that b now plays the role of the RSU-vehicle shared secret m_i in our SPECS scheme.)

Guess: Finally A sends B a bit x' as its guess for x. B answers the DDH problem positively that $T = abP$ if B's guess is correct (i.e. $x = x'$).

Now let us look at why B can answer the DDH problem in this way. If $t=0$ (i.e. $T=abP$), then $ID_2 = VPK_b \oplus H(rabP) = VPK_b \oplus H(bID_1)$ is a valid pseudo identity in proper format. In this case, since A has non-negligible advantage in the game described above, it is likely that A can break our SPECS system and can guess x correctly with probability $1/2+\epsilon$. Thus, $Pr[B\text{succeeds}|t=0] = 1/2+\epsilon$. If $t=1$, we claim that $Pr[B \text{ succeeds}|t=1] = 1/2$ only. To see why, we observe that when T is randomly chosen, the term $H(rT)$ in ID_2 cannot be cancelled by the term $H(bID_1)$ and so there is no way to obtain VPK_x. Thus the computation reveals no information about x. In this sense, the value of x is hidden to A, so even A can break our SPECS system, the probability that he will guess x correctly is simply 1/2 (by tossing a fair coin). Hence, $Pr[B$ succeeds$] = 1/2 \times (1/2 + \epsilon) + 1/2 \times 1/2 = 1/2 + \epsilon/2$. Since ϵ is non-negligible, B can solve the DDH problem but this violates the assumption that DDH is hard. Therefore, our SPECS scheme is secure in the sense that the pseudo identity of a vehicle can preserve its real identity. □

On the other hand, the random nonce r makes the pseudo identity of a vehicle different in different messages. This makes tracing the location of a particular vehicle over time difficult without the shared secret between the sender and the RSU. Furthermore, since the verification public key VPK_i of a certain vehicle is different as seen by different RSUs, even all RSUs collude, they have no way to trace a particular vehicle's travelling route.

C. Traceability and revocability

Section IV-D shows that TA is able to trace a vehicle's real identity, thus traceability is satisfied. Also TA can revoke a vehicle from future usage, thus revocability is also satisfied.

VI. Analysis on Bloom Filter Approach

This 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 probabily of having a false positive in our bloom filter apporach (i.e., case 3 in Table I) 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 $Pr(case3) \sim 2(0.6185^{\frac{m}{n}}(1-0.6185^{\frac{m}{n}}))$. Fig. 6 shows the value of $Pr(case3)$ as the ratio of $\frac{m}{n}$ varies from 1 to 10. 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. (Note that when $\frac{m}{n} = 5$ and when $\frac{m}{n} = 10$, the false positive rate of a single bloom filter are 0.39 and 0.15 respectively.) 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.

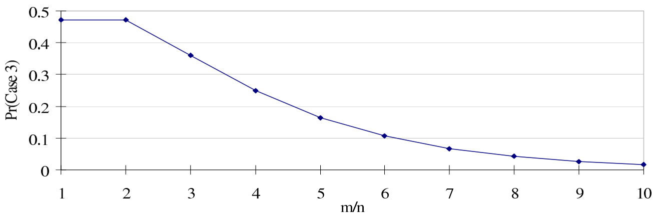

Fig. 6. Pr(case 3) with different values of m/n

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 of 16 bytes long while the H() value sent by the RSU in the notification phase is 16 bytes long per message. After that the RSU signs the notification message using an ECDSA signature which is 56 bytes long. Together with a message header of 2 bytes long, 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 of 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 long, 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. (refer to Fig. 7 for a more detailed analysis).

VII. Simulation Results

In this section, we further compare our schemes with the IBV protocol in terms of (1) the delay and (2) successful rate through extensive simulations. Note that IBV also uses a batch verification scheme, so is much faster than the RAISE protocol. Thus, we compare the delay of our scheme with the IBV [7] protocol. For successful rate, we expect we will have

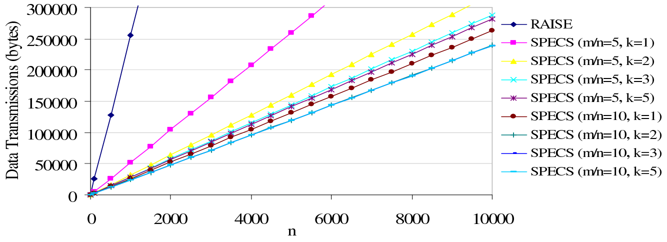

Fig. 7. Data Transmission vs. Number of Signatures in the Batch

a similar performance as RAISE as we both will identify all valid signatures even if there are invalid ones within the same batch. So, we compare our performance with the IBV protocol. Note that we only compare the schemes on handling ad hoc message. We show that our scheme can verify more signatures while the additional delay required is insignificant.

A. Simulation Models

We implement our SPECS scheme, the IBV and the RAISE protocols on a simulator written in C++. Some of the settings and parameters of our simulation are adopted from works [7] and [5]. We assume an RSU is installed on a highway and vehicles passes through it at speeds varying from 50 km/h to 70 km/h. The RVC and the IVC ranges are set to 600 m and 300 m respectively. That is, when a vehicle enters the 600 m RVC range of the RSU, the messages sent by it can received by the RSU and at the same time, the messages sent by the RSU can be received by it. Inter-vehicle messages are sent every 500 ms at each vehicle. IEEE 802.11a is used to simulate the medium access control layer. That is, when a vehicle wants to transmit, it first detects whether the channel is available. If another vehicle is transmitting, it waits until that transmission is completed and then waits for a random delay period before it begins to transmit. The bandwidth of the channel is 6 Mb/s and the average length of inter-vehicle message is 200 bytes. We assume each pairing operation takes 4.5 ms.

Our simulation runs for 1000 s. We first vary the total number of vehicles that have ever entered 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 then vary the inter-vehicle message signature error rate from 1% to 10% to interpret its impact on the performance of our schemes. Finally we vary the RSU's batch verification period from 100 ms to 1000 ms in steps of 100 ms to investigate its impact on different schemes. For each configuration, we compute the average of 5 different random scenario.

B. Simulation Results

In the first set of experiments, we assume that the RSU performs batch verification every 300 ms. We then fix the signature error rate, which is defined as the percentage of signatures which are invalid, to 5% and vary the total number of vehicles that have ever entered RSU's range throughout the simulation. Here we only consider batches that contain invalid signatures (Invalid batch). In [5], the expression for successful rate is defined. We extend its definition to handle invalid batch.

$$IBSR = \frac{1}{N}\sum_{i=1}^{N}\frac{M^{i}_{app}}{M^{i}_{mac}}$$

where M^{i}_{app} represents the total number of messages that are successfully verfied by the RSU and are consumed by vehicle i in the application layer before vehicle i leaves RSU's IVC range. Also the signatures of these messages are batch-processed with at least one invalid signature by the RSU. M^{i}_{mac}, on the other hand, represents the total number of messages received by both vehicle i and RSU in the medium access control layer from other vehicles and again the signatures of these messages are being batch-processed with at least one invalid signature by the RSU. For our schemes, we can have different levels of binary search as mentioned in Section IV. We use the notation SPECS(BSx) to denote our schemes with x levels of binary search. In Fig. 8, we can see that the IBV protocol and our schemes with single level searching (i.e. without breaking down a batch to continue the searching when the pairing on the batch failed) gives 0% invalid batch successful rate. This is because as long as there is an invalid signature in a batch, the RSU considers the whole batch as invalid and all signatures in the batch is dropped. For our schemes, it can be seen that the more the levels of binary search we do, the higher the successful rate we have. In particular, even we have 2 levels of binary search, our schemes already outperforms IBV protocol by more than 45%.

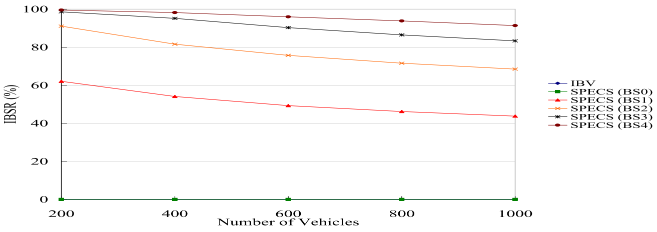

Fig. 8. Invalid Batch Successful Rate vs. Number of Vehicles

Fig. 9 shows the corresponding delay performance. We define the average delay suffered by vehicles as

$$MD = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{M}\sum_{m=1}^{M}(T^{m}_{verf} - T^{m}_{recv})$$

where M is the number of messages received by vehicle i, T^{m}_{verf} is the time that vehicle i receives the verification notification message of message m from the RSU and T^{m}_{recv} is the time that vehicle i receives message m from its neighboring vehicle. From Fig. 9, we can see that the delay under the IBV protocol and our schemes are very close to each other. For our schemes, 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 5 levels of binary search), our schemes only consume an additional of 10 ms which is roughly equivalent to the delay caused by 2 pairing operations. One more interesting point to note is that with a single level of pairing, our schemes consume 5 less ms than the IBV protocol. The reason behind is that our schemes require 2 pairing operations only while the IBV protocol requires 3 as mentioned in Section IV.

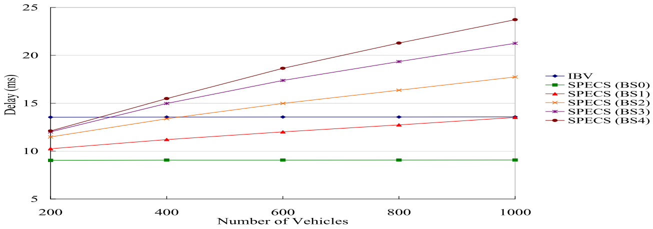

Fig. 9. Delay vs. Number of Vehicles

In this set of experiments, as the number of vehicles varies from 200 to 1000, the percentage of invalid batches increases from 13% to 50%. This makes sense because with increased number of messages, each batch contains more message signatures. Hence the probability that there is at least one invalid signature in the batch gets higher.

In the second set of experiments, we also assume that the RSU performs batch verification every 300 ms. We then fix the number of vehicles that have ever entered RSU's RVC range during the simulation period to 300 and vary the signature error rate from 0% to 10% to investigate its impact on the invalid batch successful rate and the message delay. We only consider batches that contain invalid signatures. When error rate is 0%, IBSR cannot be found as $M^i_{mac} = 0$ for all vehicles i. From Fig. 10, we see that the IBV protocol and our schemes with a single level of pairing gives 0% invalid batch successful rate due to the dropping of whole batch in case of any invalid signature. For our schemes, as we increase the levels of binary search, the successful rate becomes higher. As the error rate changes from 1% to 10%, our schemes only degrades for less that 10% and outperforms the IBV protocol for about 50%.

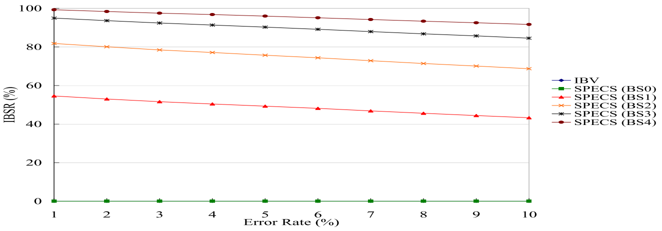

Fig. 10. Invalid Batch Successful Rate vs. Error Rate

The corresponding delay performance is shown in Fig. 11. With a single level of pairing, the delay experienced by the receiving vehicle remains constant. As discussed earlier, our schemes give a lower delay than the IBV protocol due to the save of one pairing operation. As the error rate increases, more batches contain invalid signatures. As a result, additional pairing operations are required to locate the valid signatures. This causes an increase in average delay. But the gap between our schemes and the IBV protocol is only about 10 ms.

In this set of experiments, as the signature error rate varies from 0% to 10%, the percentage of invalid batches increases from 0% to 57%. This makes sense because with higher error rate, the probability that there is at least one invalid signature in the batch gets higher.

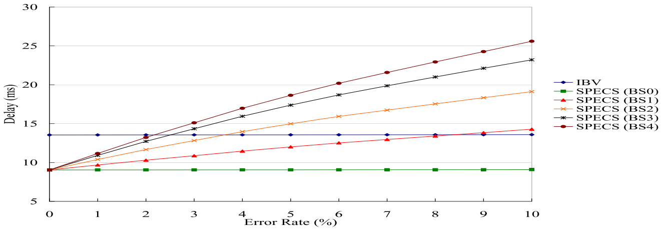

Fig. 11. Delay vs. Error Rate

In the last set of experiments, we fix the number of vehicles that have ever entered RSU's RVC range during the simulation period to 300, fix the signature error rate to 5% and vary the RSU's batch verification period from 100 ms to 1000 ms to investigate its impact on the invalid batch successful rate and the message delay. Again we only consider batches that contain invalid signatures. As shown in Fig. 12, no matter how long the RSU's batch verification period is, IBV protocol and SPECS(BS1) gives 0% invalid batch successful rate due to the dropping of whole batch in case of any invalid signature. For our schemes, as we increase the levels of binary search, the successful rate becomes higher. Further as the batch verification period changes from 100 ms to 1000 ms, our schemes degrades for about 20%. This is due to two reasons. First, a longer batch verification period implies that more vehicles' signatures are included in a batch. Thus it takes longer time to complete the binary search. Second, the RSU waits for a longer time before it starts performing batch verification. Thus a vehicle experiences longer waiting time in general. Such increase in delay is shown in Fig. 13. As a result, some vehicles may not have enough time to wait for the verification result before it leaves the concerned RSU's range.

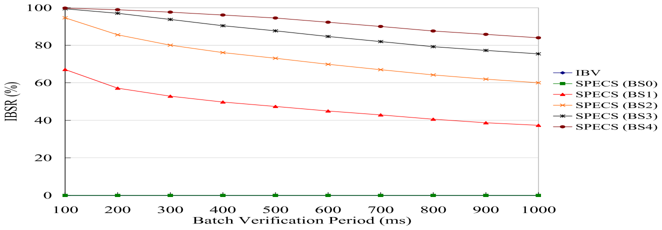

Fig. 12. Invalid Batch Successful Rate vs. Batch Verification Period

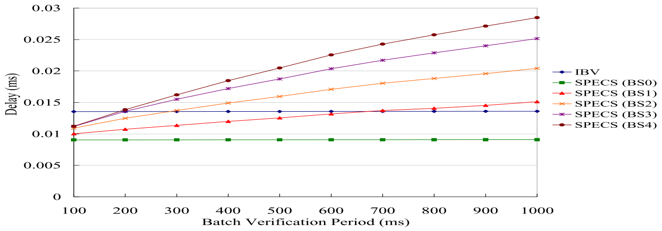

Fig. 13. Delay vs. Batch Verification Period

We magnify the range 0.01354 ms to 0.0136 ms in Fig. 14. It shows that the delay performance of the IBV protocol is also affected by the prolonged batch verification period. A vehicle in general needs to wait for a longer time before the RSU verification result is available.

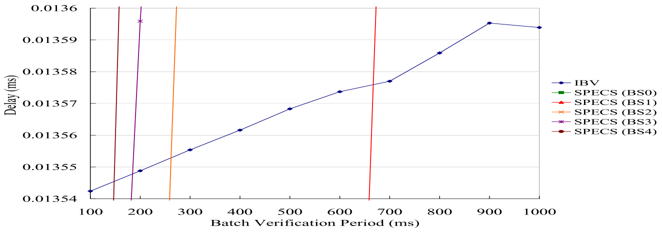

Fig. 14. Delay vs. Batch Verification Period (Magnified)

VIII. Conclusions

We proposed two secure and privacy enhancing communications schemes for VANETs to handle ad hoc messages and group messages for inter-vehicle communications. We follow the approach of letting RSU to 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 successful rate than previous works. We are also the first to propose a group communications protocol to allow known vehicles to form a group for secure communications. Note that in the early stage of VANET deployment, we may not have RSUs in all road sections. However, our protocols can be completed within the coverage of one RSU, 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. We are extending our group communications protocol to allow dynamic membership.

References

[1] F. Wang, D. Zeng, and L. Yang, "Smart Cars on Smart Roads: an IEEE Intelligent Transportation Systems Society Update," *IEEE Pervasive Computing, Vol. 5, No. 4*, pp. 68 – 69, 2006.

[2] H. Oh, C. Yae, D. Ahn, and H. Cho, "5.8 GHz DSRC Packet Communication System for ITS Services," in *Proceedings of the IEEE VTC '99*, Sept. 1999, pp. 2223 – 2227.

[3] R. Housley, W. Ford, W. Polk, and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and CRL Profile," *IETF RFC2459*, 1999.

[4] P. P. Tsang and S. W. Smith, "PPAA: Peer-to-Peer Anonymous Authentication," in *Proceedings of ACNS '08*, 2008, pp. 55 – 74.

[5] C. Zhang, X. Lin, R. Lu, and P. H. Ho, "RAISE: An Efficient RSU-aided Message Authentication Scheme in Vehicular Communication Networks," in *Proceedings of the IEEE ICC '08*, May 2008, pp. 1451 – 1457.

[6] National Highway Traffic Safety Administration U.S. Department of Transportation, "Vehicle Safety Communications Project Report," Apr. 2006.

[7] C. Zhang, R. Lu, X. Lin, P. H. Ho, and X. Shen, "An Efficient Identity-based Batch Verification Scheme for Vehicular Sensor Networks," in *Proceedings of the IEEE INFOCOM '08*, Apr. 2008, pp. 816 – 824.

[8] A. Wasef and X. Shen, "PPGCV: Privacy Preserving Group Communications Protocol for Vehicular Ad Hoc Networks," in *Proceedings of the IEEE ICC '08*, May 2008, pp. 1458 – 1463.

[9] H. Wen, P. H. Ho, and G. Gong, "A Novel Framework for Message Authentication in Vehicular Communication Network," in *Proceedings of the IEEE GLOBECOM '09*, Dec. 2009, pp. 1 – 6.

[10] A. Wasef and X. Shen, "MAAC: Message Authentication Acceleration Protocol for Vehicular Ad Hoc Networks," in *Proceedings of the IEEE GLOBECOM '09*, Dec. 2009, pp. 1 – 6.

[11] B. K. Chaurasia, S. Verma, and S. M. Bhasker, "Message broadcast in VANETs using Group Signature," in *Proceedings of the IEEE WCSN '09*, Dec. 2008, pp. 131 – 136.

[12] Y. Hao, Y. Cheng, and K. Ren, "Distributed Key Management with Protection Against RSU Compromise in Group Signature Based VANETs," in *Proceedings of the IEEE GLOBECOM '08*, Dec. 2008, pp. 1 – 5.

[13] A. Studer, E. Shi, F. Bai, and A. Perrig, "TACKing Together Efficient Authentication, Revocation, and Privacy in VANETs," in *Proceedings of the IEEE SECON '09*, June 2009, pp. 1 – 9.

[14] R. Lu, X. Lin, H. Zhu, and X. Shen, "SPARK: A New VANET-based Smart Parking Scheme for Large Parking Lots," in *Proceedings of the IEEE INFOCOM '09*, Apr. 2009, pp. 1413 – 1421.

[15] R. A. Popa, H. Balakrishnan, and A. J. Blumberg, "VPriv: Protecting Privacy in Location-Based Vehicular Services," in *Proceedings of the 18th USENIX Security Symposium*, Sept. 2009.

[16] A. Menezes, "An Introduction to Pairing-Based Cryptography," in *1991 Mathematics Subject Classification, Primary 94A60*, 1991.

[17] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," in *Proceedings of Asiacrypt '01*, 2001, pp. 514 – 532.

[18] D. Eastlake and P. Jones, "US Secure Hash Algorithm 1 (SHA1)," *IETF RFC3174*, 2001.

[19] W. Mao, "Modern Cryptography: Theory and Practice," book, July 2003.

[20] M. Bellare and P. Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols," in *Proceedings of the CCS '93*, 1993, pp. 67 – 73.

[21] D. Pointcheval and J. Stern, "Security arguments for digital signatures and blind signatures," *Journal of Cryptography, Vol. 769*, pp. 123 – 128, 2000.

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.

A. The IBV Protocol

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

1) TA chooses $\mathbb{G}$ and $\mathbb{G}$ 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 a real identity $RID \in \mathbb{G}$ and a password PWD. The drivers are informed about them during network deployment or during vehicle first registration.

When a vehicle starts up, the driver first inputs its RID and PWD into the 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 $ID_{i2} = RID \oplus H(rP_{pub1})$ 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.

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

Proof of correctness:

L.H.S.
$= \hat{e}(SK_{i1} + h(M_i)SK_{i2}, P)$
$= \hat{e}(SK_{i1}, P)\hat{e}(h(M_i)SK_{i2}, P)$
$= \hat{e}(s_1 ID_{i1}, P)\hat{e}(h(M_i)s_2 H(ID_{i1}||ID_{i2}), P)$
$= \hat{e}(ID_{i1}, s_1 P)\hat{e}(h(M_i)H(ID_{i1}||ID_{i2}), s_2 P)$
$=$ R.H.S. □

Having the pseudo identity ID_i of vehicle V_i, the TA can trace its real identity by using the *TA RID Tracing Routine*: $ID_{i2} \oplus H(s_1 ID_{i1}) = RID \oplus H(rP_{pub_1}) \oplus H(s_1 rP) = RID$.

B. 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 by following the *TA RID Tracing Routine*.

C. Anti-Traceability Attack

We describe how a vehicle can make the 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 $SK_{a2} = s_2 H(ID_{a1}||ID_{a2})$, sign the message M_a by generating the signature $\sigma_a = SK_{a1} + h(M_a)SK_{a2}$ and send out ID_a, M_a and σ_a to the RSU.

Note that the 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. The RSU forwards V_a's pseudo identity to the TA and wants it to help to reveal V_a's real identity. However, upon computing $ID_{a2} \oplus H(s_1 ID_{a1}) = GARBAGE \oplus H(rP_{pub_1}) \oplus H(s_1 rP) = GARBAGE$, the TA finds that $GARBAGE$ does not match any record at the TA. V_a can thus escape from its guity of causing the accident.

D. 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. After some while, 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, the 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. The RSU forwards V_a's pseudo identity ID_a as shown in its message to the TA and wants 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 the RSU and the 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 escape from and pass its guity of causing the accident to V_i.