SPECS: Secure and Privacy Enhancing Communications Schemes for VANETs

T.W. Chim[1], S.M. Yiu[1], L.C.K. Hui[1], Zoe L. Jiang[1], and Victor O.K. Li[2]

[1] Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong
{twchim,smyiu,hui,ljiang}@cs.hku.hk
[2] Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong
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 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. We also provide the first group communication protocol to allow vehicles to authenticate and securely communicate with others in a group of known vehicles.

Keywords: Secure vehicular sensor network, authentication, batch verification, bloom filter, group communications.

1 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.

J. Zheng et al. (Eds.): ADHOCNETS 2009, LNICST 28, pp. 160–175, 2010.

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.

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 two recent works [5, 7]. 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.

[1] Refer to [8] for the attacks.

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. None of the existing solutions provide such a protocol.

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 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. 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. 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.

2 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) 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. (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*: 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.

3 Preliminaries

Our schemes are *pairing-based* and defined on two cyclic groups with a *bilinear mapping* [9]. We briefly introduce what a bilinear map is and will discuss the basics on bloom filter which we apply in the RSU notification phase.

3.1 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.

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

4 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:

(A) **Initial handshaking:** 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:

(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 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 [7], 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.

For group messages, we have the following modules:

(E) **Group key generation:** 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.

(F) **Group message signing and verification:** 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.

4.1 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 encrypts its RID and PWD using the TA's public key PK_{TA} and sends $ENC_{PK_{TA}}(RID, PWD)$ to the RSU which forwards it to the TA.
2. The TA verifies RID and PWD. If they are valid, it generates a shared secret t_i for V_i and computes V_i's ID Verification Public Key as $VPK_i = t_i \oplus RID$. 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 sends $ENC_{PK_{TA}}(RID)$ to TA via this new RSU. 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.

4.2 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 [10]. 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 [11]. Vehicle V_i then sends $< ID_i, M_i, \sigma_i >$ to others.

4.3 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 $ID_{i2} = VPK_i \oplus H(m_i ID_{i1})$ (A more efficient approach will be presented in our full paper). It then verifies the signatures by checking if $\hat{e}(\sum_{i=1}^{n} \sigma_i, P) = \hat{e}(\sum_{i=1}^{n} m_i ID_{i1} + h(M_i)H(ID_{i2}), P_{pub})$.

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 3.2, 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 1). 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. 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.

Table 1. 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)

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 6, 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.

4.4 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 the TA and V_i itself.

4.5 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 4.2. 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.

4.6 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 4.2. 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 $\hat{e}(\sigma_i, P) = \hat{e}(ID_{i1} + h(M_i)H(ID_{i2}), GPK_i)$.

5 Analysis

5.1 Security Analysis

We analyse our schemes with respect to the security requirements listed in section 2. Formal proofs will be given in the full paper.

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 $sm_i ID_{i1}$ 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 $GPK_i = m_i P$ 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 $ID_{i2} = VPK_i \oplus H(m_i ID_{i1})$. Again, since no other vehicle knows m_i and so 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, thus cannot forge a message as well.

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.

Identity privacy preserving: The pseudo identity of any vehicle is an ElGamal-type ciphertext, which is secure under the chosen plaintext attacks [12]. Also

the random nonce r makes the pseudo identity of a vehicle different in different messages. To trace the real identity, one needs to have the shared secret between the sender and the TA. Thus no one can trace the location of a particular vehicle over time and the privacy is preserved. 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.

Traceability: Section 4.4 shows that TA is able to trace a vehicle's real identity, thus traceability is satisfied.

5.2 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 probabily of having a false positive in our bloom filter apporach (i.e., case 3 in Table 1) 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 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 the full paper for a more detailed analysis).

6 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 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 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. The simulation is performed on ad hoc messages as both other protocols do not handle group messages. We show that our scheme can verify more signatures while the additional delay required is insignificant.

6.1 Simulation Models

Some of the settings of our simulation are adopted from [7, 5]. We assume that vehicles passes through an RSU (in highway) at speeds varying from 50 km/h to 70 km/h. The RVC and the IVC ranges are set to 600m and 300m respectively. Inter-vehicle messages are sent every 500 ms from each vehicle. IEEE 802.11a is used to simulate the medium access control layer. 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 [13]. We implement the simulation using C++ language. We simulate the IEEE 802.11a protocol by generating the time stamps for broadcasting messages of each vehicle. In case two stamps collide, we randomly regenerate one of them. More details can be found in the full paper.

Our simulation runs for 1000 s. 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 interpret its impact on the performance of our schemes. For each configuration, we compute the average of 5 different random scenarios.

6.2 Simulation Results

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 [5], the expression for successful rate is defined. We extend its definition to handle invalid batch: $IBSR = \frac{1}{N}\sum_{i=1}^{N}\frac{M_{app}^{i}}{M_{mac}^{i}}$, where M_{app}^{i} is the total number of messages that are successfully verified by the RSU and are consumed by vehicle i in the application layer before vehicle i leaves RSU's RVC range, M_{mac}^{i} is the total number of messages received by both vehicle i and RSU in the medium access control layer from other vehicles. For our schemes, we can have different levels of binary search as mentioned in Section 4. We use the notation SPECS(BSx) to denote our schemes with x levels of binary search.

The successful rates for the schemes are shown in Fig. 1(a). Note that the successful rates for IBV and SPECS(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 successful rate of SPECS(BS1) is already raised to about 45%. If we raise the number of levels to 4, the successful rate can be raised to more than 90%.

While the results are quite obvious, next we will show that the delay incurred by binary search procedure is minimal. Fig. 1(b) 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 i, T_{verf}^{m} is the time that vehicle i receives the verification notification message of message m from the RSU and T_{recv}^{m} is the time that vehicle i receives message m from its neighboring vehicle.

From Fig. 1(b), 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 4 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. 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 transmission delay, so we can afford to do more pairing operations. One more interesting point to note is that without binary search, 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 4.

In the second set of experiments, we fix the number of vehicles that have entered the 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 successful rate and the message delay. We only consider batches that contain invalid signatures. Fig. 2(a) shows the results. The IBV and SPECS(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 successful rate increases.

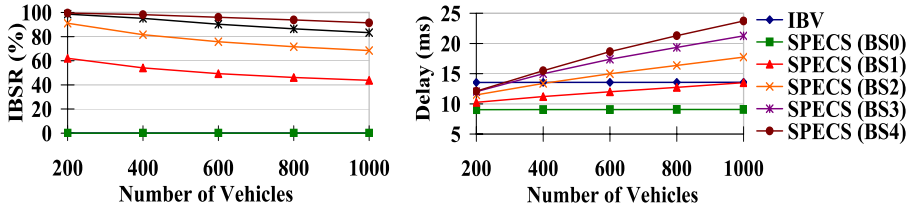

Fig. 1. (a) Invalid Batch Successful Rate vs. Number of Vehicles (b) Delay vs. Number of Vehicles

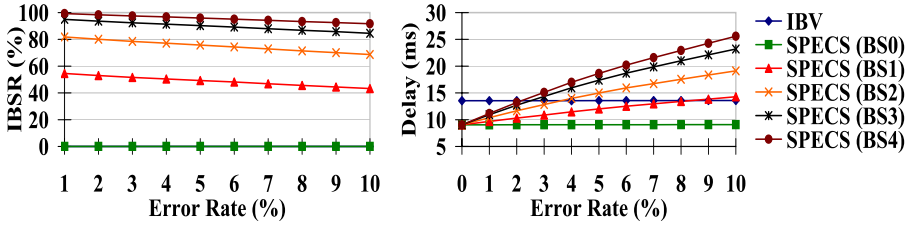

Fig. 2. (a) Invalid Batch Successful Rate vs. Error Rate (b) Delay vs. Error Rate

The interesting point is that as the error rate increases from 1% to 10%, our schemes only degrades for less that 10%.

The corresponding delay performance is shown in Fig. 2(b). As discussed earlier, SPECS(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 schemes and the IBV protocol is only about 10ms even when the error rate is 10%.

7 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. Wang, F., Zeng, D., Yang, L.: Smart Cars on Smart Roads: an IEEE Intelligent Transportation Systems Society Update. IEEE Pervasive Computing 5(4), 68–69 (2006)
2. Oh, H., Yae, C., Ahn, D., Cho, H.: 5.8 GHz DSRC Packet Communication System for ITS Services. In: IEEE Proceedings of the VTC 1999, September 1999, pp. 2223–2227 (1999)
3. Housley, R., Ford, W., Polk, W., Solo, D.: Internet X.509 Public Key Infrastructure Certificate and CRL Profile. IETF RFC2459 (1999)
4. Tsang, P.P., Smith, S.W.: PPAA: Peer-to-Peer Anonymous Authentication. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 55–74. Springer, Heidelberg (2008)
5. Zhang, C., Lin, X., Lu, R., Ho, P.H.: RAISE: An Efficient RSU-aided Message Authentication Scheme in Vehicular Communication Networks. In: IEEE Proceedings of the ICC 2008, May 2008, pp. 1451–1457 (2008)
6. National Highway Traffic Safety Administration U.S. Department of Transportation, Vehicle Safety Communications Project Report (April 2006)
7. Zhang, C., Lu, R., Lin, X., Ho, P.H., Shen, X.: An Efficient Identity-based Batch Verification Scheme for Vehicular Sensor Networks. In: IEEE Proceedings of the INFOCOM 2008, April 2008, pp. 816–824 (2008)
8. Chim, T.W., Yiu, S.M., Hui, C.K., Li, V.O.K.: Security and Privacy Issues for Inter-vehicle Communications in VANETs. In: IEEE Proceedings of the SECON 2009 (Poster Session) (June 2009)
9. Menezes, A.: An Introduction to Pairing-Based Cryptography. In: 1991 Mathematics Subject Classification, Primary 94A60 (1991)
10. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
11. Eastlake, D., Jones, P.: US Secure Hash Algorithm 1 (SHA1). IETF RFC3174 (2001)
12. Baek, J., Lee, B., Kim, K.: Secure Length-Saving ElGamal Encryption under the Computational Diffie-Hellman Assumption. In: Clark, A., Boyd, C., Dawson, E.P. (eds.) ACISP 2000. LNCS, vol. 1841, pp. 49–58. Springer, Heidelberg (2000)
13. Scott, M.: Efficient implementation of cryptographic pairings (2007), `http://ecrypt-ss07.rhul.ac.uk/Slides/Thursday/mscott-samos07.pdf`