TREES: Traceable, Revocation-Efficient, and Exculpable Signatures for Vehicular Ad-Hoc Networks

Jordan Cote, Bing Wang, Zhijie Shi, Aggelos Kiayias
University of Connecticut - Storrs, CT, USA
Hong-Sheng Zhou
Virginia Commonwealth University - Richmond, VA, USA
Technical Report: BECAT/CSE-TR-15-01

***Abstract*—Vehicular Ad-Hoc Networks (VANETs) require advanced cryptographic techniques in order to meet the conflicting requirements of simultaneous user anonymity and accountability. In this paper, we present a novel group signature scheme that meets these requirements. It provides constant-time revocation, much more efficient than what is achieved in existing schemes. It also provides exculpability so that the authority is not able to forge signatures on behalf of the users. In addition, it provides an efficient distributed tracing mechanism that does not violate the privacy of innocent users. While designed for VANETs, our signature scheme is suitable for any ad-hoc network with similar requirements.**

I. Introduction

A vehicular ad-hoc network (VANET) is formed by a large number of On-Board Units (OBUs) that are installed in vehicles and Road-Side Units (RSUs) that are deployed at road infrastructures, e.g., traffic lights, highway light poles. It provides a convenient way for the vehicles to communicate with each other and with the RSUs, which can lead to many applications for improving road safety, optimizing road traffic, and providing value added services such as entertainment and business services [1], [28], [22], [24], [19]. For these reasons, VANETs have received increasing attention from both industry [16], [20] and academia [12], [13], [14], [17], [18], [24], [27].

One challenge that must be resolved before VANETs become a widespread reality is security. Since false or malicious messages in VANETs can be extremely hazardous, messages from users must be authenticated and users must be held accountable for their messages. On the other hand, the Transit Authority (TA) must be able to reveal the identity of a misbehaving user, revoke his privileges, and "trace" all of the prior messages from the user to mitigate their damages. In addition, this tracing cannot reveal the identities of innocent users and is ideally performed in a distributed manner (e.g., by multiple Regional Authorities). Last, all the above functionalities, i.e., authentication, revocation, and tracing, must be achieved efficiently, being able to scale gracefully to the large sizes of VANETs (up to millions of users).

None of the existing approaches possesses all the above properties (see Section II). In this paper, we propose a novel group signature scheme that is traceable, revocation-efficient, and exculpable, referred to as *TREES* (Traceable, Revocation-Efficient, Exculpable Signature). Our scheme has the following desired features.

- *Constant-time revocation check based on dynamic accumulator*, which is particularly suitable for processing resource limited OBUs. As opposed to schemes that rely on RSUs for revocation (e.g., ECPP [18]), OBU credentials do not expire in our scheme, and hence the vehicular network is still usable when revocation data are out of date (e.g., at remote locations). While an existing scheme, GSIS [17], provides constant-time revocation checks, it requires the TA to divulge the private keys of revoked users. This requirement, besides opening the door to forgery attacks, also results in OBU operations that are linear in the size of revoked users since the last update. In our scheme, the equivalent of this work is offloaded to the TA, so an OBU takes constant-time to update credentials.
- *Tracing*. The tracing functionality allows the authority to discover whether any message was signed by a given user without having to *open* each signature, and hence does not compromise the privacy of innocent users. This concept was first introduced in [15]. In addition, in our scheme, multiple Regional Authorities (RAs) perform the function in a distributed way after receiving a "tracing token" from the TA. In GSIS, the term *trace* means to reveal the identity of the signer of a particular message, here named the *open* function.
- *Exculpability*. The interactive join functionality ensures that the authority is unable to forge signatures on the user's behalf, a property referred to as *exculpability*.

To the best of our knowledge, our scheme is the first VANET security scheme that proposes tracing and interactive joins. One drawback to our scheme is that it requires public key storage which is linear in the number of vehicles supported. However, this is static data that can be pre-loaded in the OBUs.

The rest of the paper is organized as follows. Section II describes related work. Section III describes the problem setting. Section IV describes preliminaries. Section V describes a group signature protocol with a constant-time revocation verification mechanism. Section VI presents our scheme for VANETs. Section VII discusses revocation. Section VIII

presents the security proof. Section IX compares our scheme with several state-of-the-art schemes. Finally, we conclude with Section X.

II. Related Work

Early studies on VANET security [24], [14] use standard public key cryptography. Since a standard signature scheme would tie a user's messages with his unique identity, thus violating user privacy, these studies achieve privacy by allowing a user to choose from a large pool of public/private key pairs that are provided by the authority and are installed on OBUs. The large number of key pairs leads to cumbersome key management and very inefficient operations for revocation.

Recent studies use group signatures as the basic cryptographic building block. GSIS [17] uses the short group signature scheme proposed in [6], and employs a hybrid accumulator/VLR (verifier-local revocation) revocation scheme. The VLR scheme is used until the revocation list size reaches a predetermined threshold. At this point the revocation list is flushed and then a new public key and private key are issued for a new epoch. The verification time is linear with respect to the revocation list size since the last epoch. This scheme meets several requirements for VANET but does not include exculpability. It also specifies that private keys of revoked users be distributed upon epoch changes. This presents a vulnerability in the ad-hoc setting where vehicles that have not yet received this information will accept forged signatures based on these private keys.

ECPP [18] uses the VLR group signature scheme proposed in [7]. After initially registering with the TA, OBUs negotiate short-time pseudonym based credentials from RSUs using their long-term credentials that are obtained from the TA. ECPP does not allow for distributed privacy-preserving message tracing. Also, this scheme's performance under heavy traffic conditions is limited because the RSUs may not be able to serve all credential requests. The RSU's other duties (facilitating dissemination of data) are hampered by the credential request traffic. Revocation is handled by simply not issuing new pseudonyms to revoked users. This indicates a dependence on the RSU for authorized communication. Since expired credentials are considered equivalent to revoked credentials, this scheme cannot be used far outside the range of an RSU.

PACP [13] is similar to ECPP and requires an OBU to obtain pseudonym token from the RSU periodically. PACP is less strenuous on the RSU by letting the OBU derive pseudonyms from the token, making the scheme more scalable. However, both of these schemes do require presence of RSUs. Otherwise, OBUs will not be able to sign messages once their pseudonym token expires. Additionally, anybody can correlate messages coming from the same pseudonym.

While not focused on VANETs, a paper by Nakanishi et al. [21] proposes a group signature protocol which has constant-time verification and does not require private key updates. Unlike the other schemes, this one also provides exculpability. However, revocation in this paper requires linear size updates. In addition, it does not include a tracing mechanism.

Other approaches for VANET security include Temporary Anonymous Certified Keys [12], [27], which use regional authorities (via RSUs) to issue temporary keys when an OBU enters a new geographical zone (or after key expires). The scheme in [27] allows short-term correlation of messages since OBUs sign under a single public key. In [12], the revocation list check is reduced to constant time using a bloom filter. However, this method is vulnerable to a non-negligible rate of false positives. In other words, signatures by valid users may be thrown out as revoked. This is not acceptable when dealing with safety messages.

III. Problem Setting

A. System model

Our system consists of a set of OBUs, a set of RSUs, a single TA (Transit Authority), and multiple RAs (Regional Authorities). An OBU is installed on a vehicle and must be registered by the vehicle's owner to the TA. Since OBUs are in motion, the amount of time they can communicate with RSUs and other OBUs is limited. The communication protocol is Dedicated Short Range Communications (DSRC) which is based on 802.11p and has an allocated 5.9 GHz band. The range of the wireless transmission is roughly 200 m and the data throughput is 6-27 Mbps. According to [28], the maximum acceptable latency is 100 ms. OBU to OBU communications are generally less than 100 bytes, and OBU to RSU communications are up to 430 bytes. OBUs are also outfitted with a high-resolution DGPS unit.

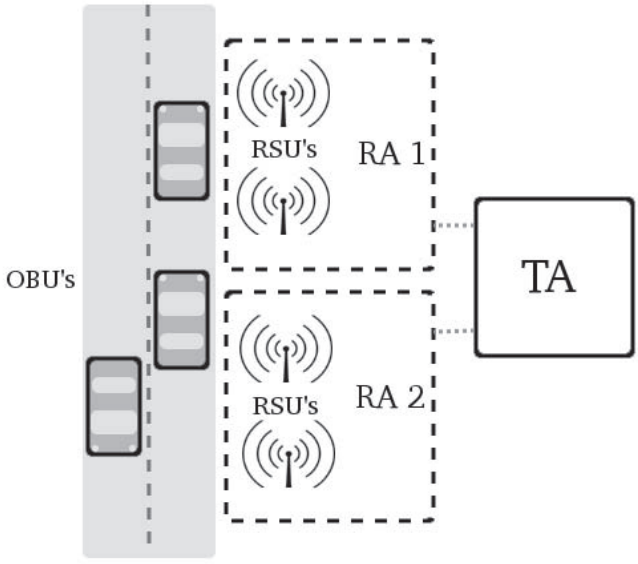

Fig. 1. System diagram showing relation between TA, RA, RSU and OBU

RSUs are fixed-location facilitator nodes that serve as proxies for relaying messages. They typically have more resources than OBUs and the communications among RSUs are efficient. Specifically, they have access to a high-speed internet connection which is used to expedite communication between OBUs separated by long distances. RSU will also

facilitate communications with the TA. These communications may include information that needs to be disseminated when vehicles are revoked. The RSUs also serve as agents of the RAs by logging transmissions. Using the logs from RSUs under its control, the RA can perform the tracing function for the TA. RSUs may also be fitted with extensions such as for RADAR imaging or traffic light controlling [28].

The TA is the central authority that manages the membership of the system, while the RAs are regional authorities at distributed locations, one in charge of a region. The TA performs functions such as registering new users and revoking signing privileges of misbehaving users.

B. Attack Model

Several attack types for VANETs have been outlined in previous studies [17], [24]:

- *Bogus Information Attack*. A user sends a signed message that contains deliberately false, misleading, malicious, and possibly hazardous information.
- *Message Replay Attack*. A user retransmits another user's signed message at a later time.
- *Message Modification Attack*. A user alters and transmits another user's signed message, such that the original signature remains valid.
- *Impersonation Attack*. A signature is forged such that the signature either opens to reveal the identity of another user, or the signature does not open properly. Here the attacker may be a user, and as a special case, the attacker may be the authority itself.
- *RSU Compromise Attack*. Results in a stolen and/or hacked RSU, which then comes under control of the attacker. The RSU may also be relocated by the attacker.
- *Movement Tracking Attack*. The attacker is able to correlate signed messages as coming from the same OBU. If the attacker can infer the vehicle's identity from a single message (for example, there are no other vehicles in range), then the attacker can track the location of that OBU.
- *ID Disclosure Attack*. The identity of a user is somehow extracted from a message signature.

C. Security Goals

We now describe the desired security features which will motivate the design of our scheme. A basic feature is that messages from an OBUs are authenticated with digital signatures which prove that the signer is a valid group member. The identity of the OBU is not revealed by the signature. Further, it is not possible to determine whether two messages were signed by the same OBU. Also, the TA should not be able to forge signatures.

The TA has the capability to "open" message signatures to reveal the identity of the original signers. The signing privileges of misbehaving users may be revoked by the TA. In addition, to mitigate damages caused by the misbehaving users, the TA is able to "trace" all of the messages they signed before revocation. This means the TA can check whether a message was signed by a given user without having to "open" it (since opening signatures may reveal the identity of an innocent user). Furthermore, "tracing" should be able to be distributed among RAs to make the operation more practical.

RSUs also have the ability to sign messages. While RSUs do not require anonymity, the TA should be able to revoke the a RSU's signing ability in case it is somehow compromised.

On top of all the above requirements, the scheme must scale gracefully to hundreds of millions of group members. This means that the running time of signing and verification algorithms is preferably constant time with respect to the size of the group. Determining whether a message's content is trustworthy is out of the scope of this paper.

IV. Preliminaries

Our proposed scheme uses group signatures as the basic cryptographic primitive. In this section, we briefly describe the concept of group signature, and several other concepts that are essential to compose our group signature scheme.

A. Group Signature

Group Signatures were first defined by Chaum and van Heyst [11]. This type of signature allows a collection of users to sign messages as a generic group member (without disclosing their exact identities). An authority is assumed in this setting. Each user obtains their own private key from the authority for signing. There is only one public key which is used to verify signatures from the entire group. The authority may also use information from the join process to "open" signatures to expose the identity of the actual signer.

The Group Signature primitive construct was formalized in [2] and for dynamic groups in [3]. In a dynamic group signature scheme, group members may be added after system initialization. Dynamic groups have a discrete join procedure rather than an at-once group instantiation. The join procedure may be interactive to provide *exculpability*, i.e., the group authority cannot forge signatures on behalf of the group members.

B. Bilinear Map

Bilinear pairing is a frequently used construct in Group Signature and will be used in our scheme. Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be multiplicative cyclic groups, all of prime order q. Let g_1 and g_2 be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. An isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ exists, and $\psi(g_2) = g_1$. It is possible that $\mathbb{G}_1 = \mathbb{G}_2$. A bilinear map is a function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that it satisfies the following properties:

- Bilinear: $\forall\ u \in \mathbb{G}_1, v \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_q$: $e(u^a, v^b) = e(u, v)^{ab}$;
- Non-degenerate: $e(g_1, g_2) \neq 1$;
- Efficiently Computable: $\exists$ an algorithm which can efficiently compute $e(u, v)\ \forall\ u, v$.

In this paper, we will sometimes use pairings such as $e(N, R)$ where $N, R \in \mathbb{G}_2$. This is done for readability, and is to be interpreted as $e(\psi(N), R)$.

C. Assumptions

Several mathematical assumptions are used in the construction of the proposed TREES group signature scheme. The first is the Diffie-Hellman Exponent (DHE) assumption [9]. It is

used to prove security of the dynamic accumulator used for revocation in this paper. A bilinear version of this assumption was first introduced in [5].

n-DHE Assumption. *Let* $\gamma \leftarrow_R \mathbb{Z}_q^*$. *Given* $\left(g_2, g_2^{\gamma^1}, g_2^{\gamma^2}, ..., g_2^{\gamma^n}, g_2^{\gamma^{n+2}}, ..., g_2^{\gamma^{2n}}\right)$, *it is computationally infeasible to output the missing* $g_2^{\gamma^{n+1}}$.

The n-Strong Diffie-Hellman assumption (n-SDH) was introduced in [4]. This assumption is used to prove that user credentials cannot be forged.

n-SDH Assumption. *Let* $x, c \in \mathbb{Z}_q^*$. *Given* $\left(g_1, g_2, g_2^x, g_2^{x^2}, ..., g_2^{x^n}\right)$, *it is computationally infeasible to produce* $\left(g_1^{1/(x+c)}, c\right)$.

The Hidden Strong Diffie-Hellman Exponent assumption (n-HSDHE) was described in [9]. It is a variation of the Hidden Strong Diffie-Hellman assumption (n-HSDH) [8]. This assumption is useful because proving knowledge of an n-HSDHE (or n-HSDH) tuple does not require using the message c directly (it is "hidden").

n-HSDHE Assumption. *Given* $f_2 \in \mathbb{G}_2$, $f_1 \leftarrow \psi(f_2)$, g_1, g_2, g_2^x, $\{g_2^{1/(x+\gamma^i)}, g_2^{\gamma^i}, f_2^{\gamma^i}\}_{i=1...n}$, *and* $\{g_2^{\gamma^i}\}_{i=n+2...2n}$, *it is computationally infeasible to produce a new tuple* $(g_1^{1/(x+c)}, g_1^c, f_1^c)$.

Finally, the Decision Linear Assumption [7] is used to prove security of revocation and signature opening features.

Decision Linear Assumption. *Let* $a, b \in \mathbb{Z}_q^*$ *and* $c \in_R \mathbb{Z}_q^*$. *Given* $u, v, h, u^a, v^b, h^c \in \mathbb{G}_2$, *it is computationally infeasible to decide if* $a+b=c$. *In other words,* h^{a+b} *is computationally indistinguishable from* h^c.

D. Dynamic Accumulator from Bilinear Pairing

The dynamic accumulator was first introduced by Camenisch et al. in [10]. An accumulator is a cryptographic primitive that essentially allows for the representation of a changing subset through a fixed-length 'accumulator' value. A value i may be added to the subset and this change is then reflected in the new accumulator value. A *dynamic* accumulator is one which allows values to be removed from the subset represented by the accumulator. A bilinear version of the dynamic accumulator was developed by Camenisch et al. in [9], which will be reviewed in this section.

One cannot infer which values are contained in a subset represented by a given accumulator. Instead, a 'witness' must be supplied to prove a value's inclusion in the subset. Given a value i, its witness, and the current accumulator, one may check if i is present in the subset. The witness must be created by the manager of the accumulator. If i is not contained in the accumulator's subset, then it is infeasible to compute such a witness.

This mechanism is well suited for revocation checking. If the signer can prove knowledge of a witness and ownership of an accumulated value, the verifier knows the signer was not revoked at the time of signing. Care must be taken to keep this proof anonymous as sending the accumulated value i would serve as a unique identifier. It is this proof that we develop into our final OBU Group Signature as well as the signature scheme for RSUs.

Accumulator Initialization. Initialization is carried out by an authority that manages the accumulator. It establishes the accumulator acc, some metadata $state$, and a master secret γ. Let the current subset of included values be V, and let the set of values that were ever added (including those which have been removed) be U. The definitions of $\mathbb{G}_1$, $\mathbb{G}_2$, g_1, g_2 and q are the same as in Section IV-B. The maximum size of the subset is n. Accumulator initialization includes setting the following values: $\gamma \leftarrow_R \mathbb{Z}_q \quad acc_\emptyset = 1 \quad z = e(g_1, g_2)^{\gamma^{n+1}}$ and $state_\emptyset = \left(\emptyset, g_2^{\gamma^1}, g_2^{\gamma^2}, ..., g_2^{\gamma^n}, g_2^{\gamma^{n+2}}, ..., g_2^{\gamma^{2n}}\right)$.

Element Accumulation. A new value is added to the accumulator by the authority. A user is given an element $B_i \leftarrow g_2^{\gamma^i}$ which is linked to the accumulated value i. Along with that, the user gets a valid witness w_i for proving accumulation. Because i is added to V and U, this process has the effect of changing the value of the public accumulator acc_V. When this happens, previously issued witnesses are no longer valid. Therefore, old witnesses need to be updated after this operation.

$$\begin{aligned} w_i &= \prod_{j \in V \setminus \{i\}} g_2^{\gamma^{n+1-j+i}} \\ acc_{V \cup \{i\}} &= acc_V \cdot g_2^{\gamma^{n+1-i}} \\ state_{U \cup \{i\}} &= \left(U \cup \{i\}, g_2^{\gamma^1}, g_2^{\gamma^2}, ..., g_2^{\gamma^n}, g_2^{\gamma^{n+2}}, ..., g_2^{\gamma^{2n}}\right) \\ wit_i &= (w_i, B_i) \end{aligned}$$

Element Removal. Revocation of a value is just the inverse of the accumulation operation. The relevant i value is removed from V, but the set U remains unchanged. wit_i is no longer valid with the new accumulator value. Witnesses need to be updated after this operation as well.

$$\begin{aligned} acc_{V \setminus \{i\}} &= \frac{acc_V}{g_2^{\gamma^{n+1-i}}} \\ state_{U \setminus \{i\}} &= state_U \end{aligned}$$

Updating Witnesses. After accumulating a new value (or removing one), the witnesses for all other users must be updated in order to be valid for the new accumulator value. Suppose the changed set is V', and the set when the witnesses were computed is V. The new witness w_i' is updated from old witness w_i. To update witness for value i, the authority checks that $i \in V'$ and that $V' \cup V \subseteq U$. The witness cannot be updated otherwise. Then the authority computes the update token Δw_i for issuance to the user. The user can then compute their new $w_i' = w_i \cdot \Delta w_i$.

$$\text{Let } \Delta w_i = \frac{\prod_{j \in V' \setminus V} g_2^{\gamma^{n+1-j+i}}}{\prod_{j \in V \setminus V'} g_2^{\gamma^{n+1-j+i}}}$$

Verification. Values can be verified as having been accumulated with respect to a particular acc value. This requires checking whether the following equality holds or not.

$$\frac{e\left(B_i, acc_V\right)}{e\left(g_1, w_i\right)} = \frac{e\left(g_1, g_2\right)^{\sum_{j\in V}\gamma^i\cdot\gamma^{n+1-j}}}{e\left(g_1, g_2\right)^{\sum_{j\in V}^{j\neq i}\gamma^{n+1-j+i}}} \stackrel{?}{=} z$$

Security. The goal of an attacker is to forge a witness w_i for a non-accumulated value $B_i = g_2^{\gamma^i}$ which corresponds to user i. We show that by producing such a wit_i the n-DHE assumption may be broken. First we derive the following relations on w_i from the verification equation.

$$e\left(B_i, acc_V\right) = e\left(g_1, w_i\right)\cdot e\left(g_1, g_2\right)^{\gamma^{n+1}} \tag{1}$$

$$e\left(g_1, \prod_{j\in V} g_2^{\gamma^{n+1-j+i}}\right) = e\left(g_1, w_i\cdot g_2^{\gamma^{n+1}}\right) \tag{2}$$

$$g_2^{\gamma^{n+1}} = \frac{\prod_{j\in V} g_2^{\gamma^{n+1-j+i}}}{w_i} \tag{3}$$

For $0 \leq i, j \leq n$, the value $(n+1) - j + i$ is always contained in $\{1, ..., n, n+2, ..., 2n\}$ and therefore the relevant value $g_2^{\gamma^{n+1-j+i}}$ is available in the public $state_U$. This means the attacker can easily compute $g_2^{\gamma^{n+1}}$ and so breaks the n-DHE assumption. However, this does not preclude the attacker from producing a valid witness for some B_i such that $i \notin V$. For this reason, we need to augment the witness wit_i to include a signature from the authority on γ^i. This signature must be checked during verification.

V. TREES: Main Functions and Cryptographic Primitives

Our scheme, TREES, is an epoch-based signature scheme. It contains two separate sets of protocols, one for OBUs and one for RSUs, since they have different requirements (see Section III). We first describe the desired functions of the Group Signature scheme. We then describe the cryptographic primitives for the OBU and RSU signature schemes, respectively. The actual constructions of the schemes are deferred to Section VI.

A. Main Functions

Setup. Let n be the maximum potential number of users in the group (OBUs and RSUs). The TA initializes the system by defining groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ as in Section IV-B. The TA then picks randomly the following generators $d_2, f_2, g_2 \in \mathbb{G}_2$. Then it sets $d_1 \leftarrow \psi(d_2)$, $f_1 \leftarrow \psi(f_2)$, $g_1 \leftarrow \psi(g_2)$. Furthermore, it chooses secrets $x, \gamma \leftarrow_R \mathbb{Z}_q$. It also creates a public/private key pair (pk, sk) and publishes pk. A hash function is chosen $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ and $H_0 : \{0,1\}^* \rightarrow \mathbb{G}_2$ (We refer the reader to Section 1.3.3 of [26] for details on implementing H_0). Last, the TA initializes the accumulator as in Section IV-D and sets $\delta = 0$. The group public key is defined as $gpk \leftarrow (g_1, g_2, d_2, f_2, n, \psi, e, H, H_0, \delta, acc_V)$. If desired, let multiple instances of $acc_{V_1}, acc_{V_2}, ...$ correlate to distinct groups of users and be used throughout the scheme in place of acc_V. Groups may differentiate between OBUs, RSUs, and sets of permissions. The signatures will need to denote which accumulator to use with an indicator, $\kappa = 1, 2, ...$, which is used in a similar manner to δ.

Epoch Change. The epoch information describes the state of the system at a given time. Changes to gpk are withheld until an Epoch Change is issued. At this time, the new gpk is published and replaces the old one. All joins and revocations since last Epoch Change are reflected in the new gpk. The only values of gpk that change are δ and acc_V. Upon Epoch Change, all users must update their private keys as well. The TA is responsible for sending updates to gpk and to individual private keys. The TA signs each user's update message using pk.

Join. A user is admitted into the group and assigned an identity value i which has not been already issued. A private key is established for the user as a product of interaction with the TA. The user gives the TA a value D_i for which the discrete log is only known to the user. Note that, although i has not been chosen yet, we use the notation D_i for clarity. A secure channel must be established for this exchange between TA and the user. This secure channel may make use of pk, sk. After registration, acc_V is updated. The TA privately records tracing token, A_i, for the user along with its identity. Finally, the user receives the private key gsk_i from the TA.

Revoke. A user is removed from the group. This prevents their signatures from being verifiable after the next epoch change takes effect. The public key gpk is updated.

Sign. Group member, i, uses its private key to produce a signature, σ, for message m, which convinces a verifier that the message originated from a current (anonymous) group member.

Verify. The authenticity of the signature is verified by checking the signature using gpk of the current epoch.

Update. A user checks with the TA whether it is up-to-date with the current epoch δ. If it is not, the user sends a request for update data. With that, the user updates their private and public key. It may keep the old public key for verifying messages from out-of-date users.

Open. The TA takes a signature σ and uses information saved from **Join** to output the identity of the actual signer.

Issue Tracing Token. Given a user i, the TA produces a tracing token A_i for distribution to the RAs.

Trace. Using token A_i and a group signature σ, a verifier (typically an RA) determines whether or not user i created the signature. This could be considered a more restrictive version of **Open**.

B. Signature Scheme for OBUs

The signature scheme for OBUs is a group signature scheme with the dynamic accumulator serving as the basis. For each group member, its accumulator witness will be used as part of its credentials. As noted at the end of Section IV-D, the witness should be accompanied by a signature from the TA

referred to as the *credential signature*. For group member i to actually sign a message, it must prove knowledge of the value w_i as well as the *credential signature* in zero knowledge. Finally, the knowledge proof is converted into a set of protocols to be used for signing and verifying. Using the dynamic accumulator, we achieve constant-time verification, including revocation checks, without publishing any private keys. We also modify the protocol to have interactive joins for exculpability (Section V-C). A tracing function is defined, which allows a token issued from the authority to be used to tell whether or not a signature is from a particular user. The tracing tokens may also be used for emergency revocation checks.

1) Protocol for Creating Credential Signature

The first part of constructing the OBU group signature is describing the **Join** procedure. To begin, the user selects a value $y_i \leftarrow_R \mathbb{Z}_q$ and sends $D_i \leftarrow d_2^{y_i}$ to the TA. The TA then issues a witness and credential signature to the user. The witness generation process was described in Section IV-D. The credential signature scheme is based on one in [9]. This scheme, however, also provides exculpability by keeping some information from the TA.

In the group signature setting, the verification cannot reveal any information about i, which could identify the user. So a proof of accumulation of a hidden value is required. A simple approach would be to use a disjunction of proofs for all valid i's, but in this case the size and verification time of signatures increases linearly with the group size. Instead, we use a simpler signature scheme that only signs messages of the form γ^i. The signature is such that the value $g_2^{\gamma^i}$ is used to verify, making the connection between the signature and the proof of accumulation easier. All that is left is to show that the according value has been accumulated. This is done by randomizing the values in the verification equation. Randomization is done by using Pedersen commitments [23] of sensitive values and satisfying equations based on those values.

When signing a message, the user must prove ownership of an accumulated value. More specifically, the user proves knowledge of the following:

- A signature (A_i, B_i, C_i) on value γ^i from the TA.
- That same γ^i was accumulated in acc_V.
- A value y_i, where $d_2^{y_i}$ was used by the TA to create the signature.

This gives an idea what the requirements are for the TA's credential signature protocol which creates the *credential signatures* for issuance to group members. The signature scheme used is based on the n-HSDHE assumption. Here, it is extended in order to achieve exculpability using an additional input D_i. This technique was described for plain n-SDH in [7].

Protocol 1 which creates a credential signature for a group member is summarized in Fig. 2. We end up with a signature (A_i, B_i, C_i) that satisfies $g = A_i^{sk+\gamma^i} \cdot D_i$. Essentially, this is a signature by the TA on the stationary value γ^i which has been accumulated. This relation is checked by the verifier using the bilinear pairing.

generators $d_2, f_2, g_2 \in \mathbb{G}_2$
set $d_1 \leftarrow \psi(d_2), f_1 \leftarrow \psi(f_2), g_1 \leftarrow \psi(g_2)$
given $D_i = d_2^{y_i}$
secret key $sk \in \mathbb{Z}_q$
public key $pk = g_2^{sk}$
return $\left(A_i = \left(\frac{g_2}{D_i}\right)^{\frac{1}{sk+\gamma^i}}, B_i = g_2^{\gamma^i}, C_i = f_2^{\gamma^i}\right)$
verify $e(pk \cdot B_i, A_i) \cdot e(g_1, D_i) \stackrel{?}{=} e(g_1, g_2)$
and $e(B_i, f_2) \stackrel{?}{=} e(g_1, C_i)$

Fig. 2. (Protocol 1) Create credential signature for an OBU.

2) TREES Knowledge Proof for OBUs

After the accumulator is initialized as in Section IV-D, user i is issued signing credentials as in the previous section. At this time, the relevant γ^i is accumulated. Once the user has the credential set $(A_i, B_i, C_i, y_i, w_i)$ they may prove group membership. This allows them to create group signatures. The signer begins the protocol by sending the commitments described below. We require new generators $h_2, \ell \in \mathbb{G}_2$, whose discrete logarithms along with g_2 are mutually unknown. Let $h_1 \leftarrow \psi(h_2)$. We also define $a \leftarrow \mathbb{Z}_q^*$. The knowledge proof requires some method of randomizing the secret values. We use Pedersen commitment to commit to the secret values as shown below. We select values $t_1, t_2, t_3, t_4, t_5, t_6 \leftarrow_R \mathbb{Z}_q$ and set the following:

$$G = B_i h_2^{t_1} \qquad T = g_2^{t_1} h_2^{t_2} \qquad F = C_i h_2^{t_3} \qquad S = A_i h_2^{t_4}$$
$$W = w_i h_2^{t_5} \qquad Y = D_i h_2^{t_6} \qquad L = \ell^{t_4}$$

Now we are ready to describe the knowledge proof. In the following notation, items in the parenthesis are secret values known to the prover. All other values are also known to the verifier as well. The rest in curly brackets are equations to be satisfied based on the commitments.

$$\begin{aligned}
\mathbf{PK}\{(t_1, t_2, t_3, t_4, t_5, t_6, mult, tmp, y_i) : \\
T &= g_2^{t_1} h_2^{t_2} \quad \wedge \\
1 &= T^{t_4} g_2^{-mult} h_2^{-tmp} \quad \wedge \\
\frac{e(pk \cdot G, S) \cdot e(g_1, Y)}{e(g_1, g_2)} &= e(pk \cdot G, h_2)^{t_4} \cdot e(g_1, h_2)^{t_6} \\
&\quad \cdot e(h_1, h_2)^{-mult} \cdot e(h_1, S)^{t_1} \quad \wedge \\
\frac{e(G, acc_V)}{e(g_1, W) z} &= e(h_1, acc_V)^{t_1} e(g_1, h_2)^{-t_5} \quad \wedge \\
\frac{e(G, f_2)}{e(g_1, F)} &= e(h_1, f_2)^{t_1} e(g_1, h_2)^{-t_3} \quad \wedge \\
Y &= d_2^{y_i} h_2^{t_6} \quad \wedge \quad L = \ell^{t_4}\}
\end{aligned}$$

The first equality proves knowledge of t_1. The second equation sets $mult = t_1 \cdot t_4$ and $tmp = t_2 \cdot t_4$. This is used in the third which establishes $e(pk \cdot B_i, A_i) \cdot e(g_1, D_i) = e(g_1, g_2)$. The fourth ensures $\frac{e\left(g_2^{\gamma^i}, acc_V\right)}{e(g_1, w_i)} = z$. The fifth says that

$e(B_i, f_2) = e(g_1, C_i)$. The last equation proves that the discrete logarithm t_4 of L with base ℓ is the same as the one used in the commitment to A_i. All of these properties are necessary and sufficient to prove group membership, and therefore the PK statement is satisfactory for a group signature (completeness & soundness).

3) TREES Group Signature Protocol for OBUs

The implementation of the knowledge proof, referred to as **Protocol 2**, is shown in Fig. 3. It is a modified Schnorr (sigma) protocol [25]. If all verification equations hold, then the signature is accepted as originating from a valid (non-revoked) member of the group. However, the user's identity is not revealed due to the zero-knowledge proof[1]. The following lemmas show that our protocol is a zero knowledge proof which provides exculpability; their proofs are found in Appendices A, B, C and D respectively.

Lemma 1. *Protocol 2 is complete.*

Lemma 2. *Transcripts of Protocol 2 can be simulated.*

Lemma 3. *There exists an extractor for Protocol 2.*

Lemma 4. *Protocol 2 provides exculpability based on knowledge of y_i.*

C. Tracing

Tracing reveals whether or not the signature was issued by a particular user. The implementation is based on the Decision Linear assumption. A signature may be identified using the tracing token A_i:

$$e\left(\frac{S}{A_i}, \ell\right) \stackrel{?}{=} e\left(h_1, L\right) \quad \Rightarrow \quad e\left(h_1^{t_4}, \ell\right) \stackrel{?}{=} e\left(h_1, \ell^{t_4}\right)$$

D. Signature Scheme for RSUs

The signature scheme for RSUs is actually not a group signature since anonymity is not required. However, we do require an efficient revocation, so we design a protocol similar to that of the OBUs.

Since traceability and exculpability are not needed for RSUs (since RSUs do not require anonymity and they are supposedly owned by the TA), we use the signature based on the n-HSDHE problem with no further modifications to create credential signature for RSUs (referred to as **Protocol 3**), as summarized in Fig. 4.

generators g_1, g_2, f_1, f_2
secret key $sk \in \mathbb{Z}_q$
public key $pk = g_2^{sk}$
return $\left(A_i = g_2^{\frac{1}{sk+\gamma^i}}, B_i = g_2^{\gamma^i}\right)$
verify $e(pk \cdot B_i, A_i) \stackrel{?}{=} e(g_1, g_2)$

Fig. 4. (Protocol 3) Create credential signature for an RSU.

[1]Actually, this proof is not completely zero-knowledge because it allows the authority to reveal the user's identity in special circumstances.

The RSU signature scheme only requires commitments W, S from before. B_i may be given to the verifier. The knowledge proof is also simplified, compared to the OBU's.

$$\begin{aligned}&\mathbf{PK}\{(t_4, t_5):\\ &\frac{e(pk \cdot B_i, S)}{e(g_1, g_2)} = e(pk \cdot B_i, h_2)^{t_4} \quad \wedge\\ &\frac{e(B_i, acc_v)}{e(g_1, W) \cdot z} = e(g_1, h_2)^{-t_5} \quad \}\end{aligned}$$

Last, the TREES signature protocol for RSUs (referred to as **Protocol 4**) is summarized in Fig. 5.

VI. TREES Group Signature Scheme Construction

This section describes the various functions of the TA, OBU and RSU. We also describe the functions which are shared by all entities.

A. TA Functions

Setup. The TA initializes the system as in **Setup** in Section V-A.

Epoch Change. TA publishes epoch data $epoch_{\delta+1} \leftarrow (\delta + 1, acc_V)$ for the new set of valid users V. This is signed using sk. The TA must then issue update tokens Δw_i for each user i which was not revoked ($i \in V$). The token Δw_i is computed as in Section IV-D.

Revoke. The TA revokes a user (OBU or RSU) i by updating the accumulator values: $acc_{V \setminus i} \leftarrow acc_V \cdot g^{-\gamma^{n+1-i}}$ and $state_{U \setminus i} \leftarrow state_U$. The changes will go into effect after the next epoch change.

Trace. To trace OBU user i, the TA looks up A_i and issues it to the RAs (Regional Authorities). The RAs act as tracing deputies which run the tracing check as in Section V-C. The results of the trace (all of the messages found which were signed by OBU i) are aggregated back to the TA.

Open. For signature σ, the TA performs the tracing check for each A_i until a match is found. Then, the TA outputs the identity associated with the matching A_i.

B. OBU Functions

Join. When an OBU wants to register as a group member, a secure channel must first be established. This requires an established a public key infrastructure or similar setup. Note that although i is assigned in step 3, we will refer to it from the start for simplicity of notation. The TA registers a new OBU as follows:

1: User i chooses $y_i \leftarrow_R \mathbb{Z}_q$.
2: User i computes and sends $D_i \leftarrow d_2^{y_i}$ to the TA.
3: TA chooses available $i \in [1..n]$.
4: TA sets $acc_V \leftarrow acc_{V \cup \{i\}}$,
$state_U \leftarrow state_{U \cup \{i\}}$,
and w_i (as in Section IV-D).
5: TA updates witnesses $w'_j \leftarrow w_j \cdot g_2^{\gamma^{n+1-i+j}}$.
for each $j \in V, j \neq i$ (as in Section IV-D).

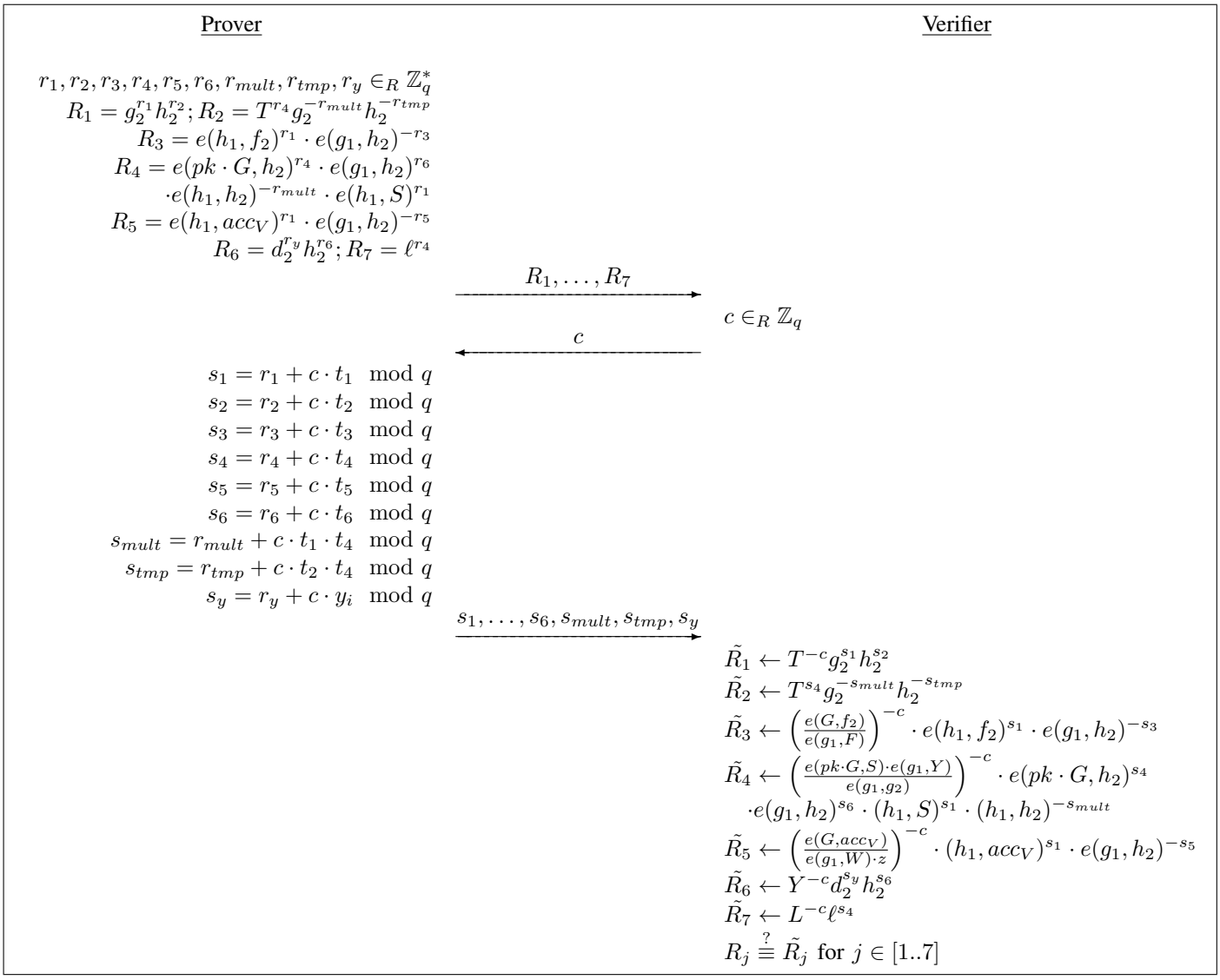

Fig. 3. (Protocol 2) TREES Group Signature Protocol for OBUs.

6: TA computes A_i, B_i, C_i as is Protocol 1.
7: TA gives (w_i, A_i, B_i, C_i) to user i.
8: TA signs and publishes
$epoch_{\delta+1} \leftarrow (\delta + 1, acc_V)$.
9: User i verifies w_i as in Section IV-D.
10: User i verifies (A_i, B_i, C_i) as in Protocol 1.

Sign. Given a message $m \in \mathbb{Z}_q$, an OBU may sign as follows:

1: User i chooses $\tilde{r} \leftarrow_R \mathbb{Z}_q$.
2: User i computes $(h_2, \ell) \leftarrow H_0(gpk, m, \tilde{r})$.
3: User i computes
$G, T, F, S, W, Y, L, R_1, R_2, R_3, R_4, R_5, R_6, R_7$
as in Protocol 2
4: User i computes $c \leftarrow H(m, \tilde{r}, \delta,$
$G, T, F, S, W, Y, L, R_1, R_2, R_3, R_4, R_5, R_6, R_7)$.
5: User i computes
$s_1, s_2, s_3, s_4, s_5, s_6, s_{mult}, s_{tmp}, s_y$
as in Protocol 2.
6: User i sets $\sigma \leftarrow (G, T, F, S, W, Y, L, \tilde{r}, c,$
$s_1, s_2, s_3, s_4, s_5, s_6, s_{mult}, s_{tmp}, s_y, \delta, \kappa)$.

Using a 170 bit prime for q where elements in $\mathbb{G}_1$ are 171 bits, the size of σ is 3067 bits, plus the size of δ and κ.

Update. The OBU checks with an RSU if the local δ value is current. If not, the OBU sends a request to the TA (via the RSU) for an update. The request contains the user's B_i value and the OBU's current δ. The request data is encrypted under the TA's public key. The TA looks up the user record using a hash map and returns update information for each epoch since the OBU's current δ. Each epoch update contains the values δ, acc_V, and Δw_i. Upon receiving the update, OBU verifies the TA's signature on the update data. OBU then computes its new w_i as $w_i \cdot \Delta w_i$ and retains the other values to use for verification.

C. RSU Functions

Join. The TA registers a new RSU as follows:

1: TA chooses available $i \in [1..n]$.

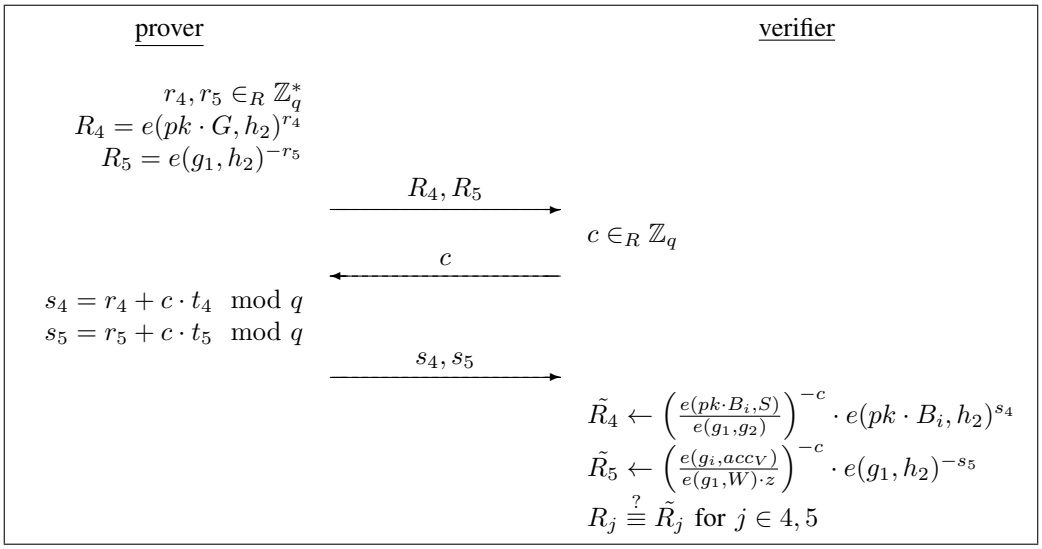

Fig. 5. (Protocol 4) TREES Group Signature Protocol for RSUs.

2: TA sets $acc_V \leftarrow acc_{V \cup \{i\}}$, $state_U \leftarrow state_{U \cup \{i\}}$, and w_i as in Section IV-D.
3: TA updates witnesses $w'_j \leftarrow w_j \cdot g_2^{\gamma^{n+1-i+j}}$ for each $j \in V, l \neq i$ as in Section IV-D.
4: TA computes A_i, B_i as in Protocol 3.
5: TA gives (w_i, A_i, B_i) to RSU i.
6: TA signs and publishes $epoch_{\delta+1} \leftarrow (\delta + 1, acc_V)$.
7: RSU i verifies w_i as in Section IV-D.
8: RSU i verifies (A_i, B_i) as in Protocol 3.

Sign. Given a message $m \in \mathbb{Z}_q$, the RSU creates a signature as follows:

1: RSU i computes S, W, R_4, R_5 as in Protocol 4.
2: RSU i computes $c \leftarrow H(m, \tilde{r}, \delta, S, W, B_i, R_4, R_5)$.
3: RSU i computes s_4, s_5 as in Protocol 4.
4: RSU i sets $\sigma \leftarrow (S, W, B_i, \tilde{r}, c, s_4, s_5, \delta, \kappa)$. Using a 170 bit prime for q where elements in $\mathbb{G}_1$ are 171 bits, the size of σ is 1193 bits, plus the size of δ and κ.

Update. The RSU sends a request to the TA for an update. The request contains the RSU's B_i value and the RSU's current δ. The rest of the procedure is the same as the OBU's Update function.

D. Shared Functions

Verify (OBU)**.** To verify an OBU signature, the verifier first computes $\tilde{R_1}, \tilde{R_2}, \tilde{R_3}, \tilde{R_4}, \tilde{R_5}, \tilde{R_6}$. It then carries out verification checks as in Protocol 2.

Verify (RSU)**.** For verification of an RSU signature, the verifier computes $\tilde{R_4}, \tilde{R_5}$ and performs verification checks as in Protocol 4.

VII. Revocation

The signatures of the OBU and RSU not only prove group membership but also that the signer has not been revoked. The revocation status verification is current as of the start of the epoch associated with the signature. The interval length between epoch changes determines the confidence that a signer is not currently revoked. In TREES, the epoch update data is small and efficient to process for the OBU and RSU. So, the update interval is primarily limited by the TA's processing ability to compute new Δw_i for each user i. Computation of Δw_i has a run time of $O\left(n_{added} + n_{revoked}\right)$ where n_{added} is the number of vehicles registered since the last epoch, and $n_{revoked}$ is the number of vehicles revoked since the last epoch. The TA may choose to pre-compute Δw_i for each user, or compute them on-demand. Also note that an epoch update may be issued in regular intervals, but also can be pushed immediately if there is a user that must be added or removed.

There may be certain cases where it would be desirable that the verifier is able to discern the identity of a revoked signer. This would be useful in cases where the revoked user is particularly malicious or dangerous. In this case, the tracing tokens of Section V-C may be distributed to all verifiers in addition to the RAs. This usage is discouraged except for only exceptional cases since the running time is linear in the number of tracing tokens to check.

VIII. Security

In this section we show how TREES addresses the various attacks specified in Section III-B.

- *Bogus Information Attack*. This is thwarted by the **Open Trace** and **Revoke** abilities of the TA, which hold the user accountable for their signatures.
- *Message Replay Attack*. Addressed by the inclusion of δ in the computation of the signature σ. It means the verifier will be able to see which epoch the signature was created under, and may decide to ignore old messages. For additional security, a timestamp may be included in the message.
- *Message Modification Attack*. Due to the inclusion of the message in the computation of the signature, this attack is not feasible.
- *Impersonation Attack*. To impersonate a user i, the attacker needs to forge the credential signature which

means they can break the n-HSDHE assumption. The form of the knowledge proof ensures that the A_i used in the tracing token is the same as the A_i used to prove group membership. The group signature uses the knowledge of the credential signature and the validity of witness w_i. As described in Section IV-D, the ability to forge a verifiable w_i is equivalent to breaking n-DHE. Also, Lemma 4 shows that if the TA can forge a signature without y_i, it can break the discrete log assumption.

- *RSU Compromise Attack*. Once the RA notices that an RSU is missing or misbehaving, the TA may revoke the affected RSU.
- *Movement Tracking*. OBU signature proofs are zero-knowledge unless the attacker knows A_i, which is only known to the TA. This fact also makes the *ID Disclosure Attack* impossible as well.

IX. Performance Evaluation

Aside from the features of an authentication system, an important aspect to consider is the efficiency. We evaluate the efficiency of TREES by comparing the run time of signing and verification, the size of a signature, and the size of the revocation list updates of TREES with several state-of-the-art schemes. In addition, we investigate the impact of the extra processing time due to verification on network performance using realistic simulations.

A. Computation and Communication Costs

Table I compares the computation costs and complexities of the proposed scheme with several state-of-the-art schemes. It shows that our scheme achieves all the desired properties with compatible computation costs and complexities as other schemes. In addition, our scheme also has very low communication cost in updating the revocation list: our revocation list update data consists of just acc_V and Δw_i which can be distributed from RSUs to OBUs quickly as they pass by. Since elements in $\mathbb{G}_1$ are 171 bits each, the revocation list update is only 43 bytes.

TABLE I
COMPARISON OF TREES WITH SEVERAL STATE-OF-THE-ART SCHEMES BY RUN TIME AND SIGNATURE SIZE. P, E DENOTE THE NUMBER OF PAIRING AND EXPONENTIATION OPERATIONS, RESPECTIVELY. $|\delta|$ IS THE SIZE OF THE VALUE δ IN BYTES, AND $|RL|$ IS THE NUMBER OF REVOKED VEHICLES.

	TREES	GSIS	ECPP	[21]
OBU Sign	8 P, 24 E	1 P, 12 E	1 E	14 E
OBU Verify	14 P, 22 E	$2+3\|RL\|$ P, 12 E	3 P	4 P, 11 E
RSU Sign	2 P, 4 E	2 E	–	–
RSU Verify	5 P, 4 E	1 P, 2 E	–	–
OBU Sig.	$384B + \|\delta\|$	$192B$	$189B$	$650B$
RSU Sig.	$150B + \|\delta\|$	$83B$.	–	–
RL Size	$43B$	Limited Linear	0	Linear

B. Verification Processing Speed

For transmission from one OBU to another OBU, the end-to-end delay will be increased due to verification. Additional processing time due to verification may cause buffer overflow and packet loss. In our simulations, we find that the delay due to verification processing is less than 20ms in the worst case. This delay is acceptable, so we focus on packet loss in this section.

The VanetMobiSim tool [29] creates realistic traffic movement patterns based on US Census TIGER data. In particular, we use a 1000 x 1000 meter span of suburban roads in Fairfield County, Connecticut. Vehicles are randomly placed on intersections and instructed to move to another random intersection. Vehicles move according to typical traffic patterns, obeying speed limits and stopping momentarily at intersections.

Network communication is simulated using ns-2. All vehicles send broadcast packets with inter-packet sending times following a Normal distribution with a mean of 1800ms and a variance of 400ms. Communication range is approximately 200m using a two-ray ground propagation model. The number of vehicles in the experiment ranges from 20 to 100. Simulations last for 300s. Each experiment is repeated three times and we obtain the average from the multiple runs. Also, experiments of this kind have a warm-up period prior to stabilization of loss rates. The length of this period varied according to the parameters and the maximum length was 5000s. Therefore, we start the 300s measurement after 5000s of warm-up in all experiments.

The verification times were computed using the numbers in Table I. We assume the running time of 4.5ms for pairing and 0.6ms for exponentiation operations, as specified in [18].

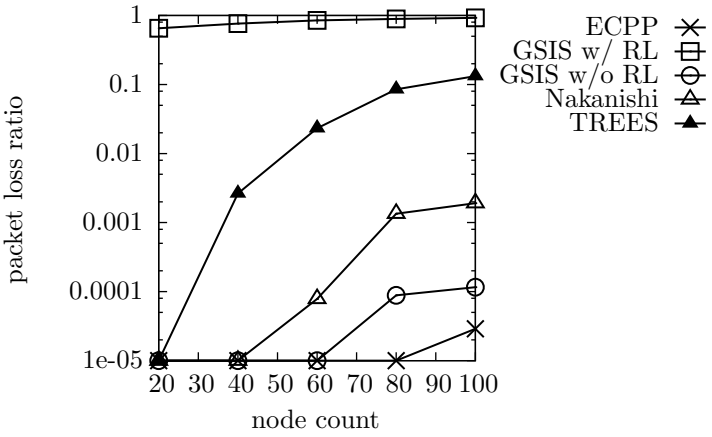

Fig. 6. Loss rates using various authentication protocols.

Figure 6 plots the loss rates of several schemes as the number of nodes varies from 20 to 100. The scheme with the lowest packet loss rate is ECPP, due to its simple revocation mechanism. This scheme has additional overhead from pseudonym requests not accounted for in this experiment. This mechanism also has a security problem as described in Section II. GSIS with an empty revocation list (GSIS w/o RL) also has low loss rates. But since the size of RL is always growing between epoch intervals, it does not represent the average loss rate. GSIS w/ RL represents the loss rate of GSIS where $|RL| = 100$. The loss ratio here ranges from 65.4% to 92.3%. Since VANETs can contain up to millions of nodes, epoch updates would have to be very frequent to maintain a threshold of less than 100 nodes. Therefore, we find this approach impractical. The loss rate of the scheme proposed

by Nakanishi et al. is also low. However, this scheme has a linear revocation list update size and does not support tracing.

In TREES, loss rates for 20 and 40 nodes are negligible. The loss rates are 2.6%, 8.5% and 13.2% when the number of vehicles is 60, 80 and 100, respectively. Note that the loss rates of all the constant-time verification schemes, including TREES, can be reduced by increasing the buffer size. For comparison purposes, we set the buffer size of each OBU to be 6 packets. A more realistic buffer size is orders of magnitude larger, which reduces the loss rates to a negligible value for all the settings we investigate. Given this, we find packet loss rates to be acceptable for real-world use.

X. Conclusion

In this paper, we have designed TREES, a novel group signature scheme for vehicular ad-hoc networks. It provides constant-time revocation check based on the dynamic accumulator. TREES also has good computation and communication complexity in consideration with other schemes. It does not require disseminating private keys, and credentials do not need to expire. Drawbacks include large public key data (which can be preloaded) and linear time *open* operation (which can be distributed). To the best of our knowledge, TREES is the first VANET security scheme that proposes tracing and interactive joins for enhanced privacy.

References

[1] F. Bai, H. Krishnan, V. Sadekar, G. Holl, and T. Elbatt. Towards characterizing and classifying communication-based automotive applications from a wireless networking perspective. In *Proc. of IEEE AutoNet*, 2006.

[2] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Proc. of EUROCRYPT*, pages 614–629. 2003.

[3] M. Bellare, H. Shi, and C. Zhang. Foundations of group signatures: The case of dynamic groups. In *Proc. of CT-RSA*, pages 136–153. 2005.

[4] D. Boneh. Short signatures without random oracles. In *Proc. of EUROCRYPT*, 2004.

[5] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Proc. of EUROCRYPT*, pages 440–456. 2005.

[6] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. pages 41–55, 2004.

[7] D. Boneh and H. Shacham. Group signatures with verifier-local revocation. In *Proc. of ACM CCS*, 2004.

[8] X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. In *Public Key Cryptography*, pages 1–15. 2007.

[9] J. Camenisch, M. Kohlweiss, and C. Soriente. An Accumulator Based on Bilinear Maps and Efficient Revocation for Anonymous Credentials. *Public Key Cryptography*, pages 481–500, 2009.

[10] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Proc. of CRYPTO*, pages 61–76, 2002.

[11] D. Chaum and E. Van Heyst. Group signatures. In *Proc. of EUROCRYPT*, pages 257–265, 1991.

[12] J. Haas, Y.-C. Hu, and K. Laberteaux. Efficient certificate revocation list organization and distribution. *IEEE Journal on Selected Areas in Communications*, 29(3):595–604, 2011.

[13] D. Huang, S. Misra, M. Verma, and G. Xue. PACP: An Efficient Pseudonymous Authentication-Based Conditional Privacy Protocol for VANETs. *IEEE Transactions on Intelligent Transportation Systems*, 12(3):736–746, 2011.

[14] J. Hubaux, S. Capkun, and J. Luo. The security and privacy of smart vehicles. *IEEE Security & Privacy magazine*, pages 49 – 55, 2004.

[15] A. Kiayias and Y. Tsiounis. Traceable signatures. *Proc. of EUROCRYPT*, pages 1–35, 2004.

[16] KVH Industries, Inc. `http://www.kvh.com/`.

[17] X. Lin, S. Member, X. Sun, P.-h. Ho, X. S. Shen, and S. Member. GSIS: A Secure and Privacy-Preserving Protocol for Vehicular Communications. *IEEE Transactions on Vehicular Technology*, 56(6):3442–3456, 2007.

[18] R. Lu, X. Lin, H. Zhu, P.-H. Ho, and X. Shen. Ecpp: Efficient conditional privacy preservation protocol for secure vehicular communications. In *Proc. of IEEE INFOCOM*, pages 1229–1237, 2008.

[19] C. Merlin and W. Heinzelman. A study of safety applications in vehicular networks. In *Proc. of IEEE MASS*, 2005.

[20] MSN TV. `http://www.msntv.com/`.

[21] T. Nakanishi, H. Fujii, Y. Hira, and N. Funabiki. Revokable Group Signature Schemes with Constant Costs for Signing and Verifying. *Public Key Cryptography*, pages 463–480, 2009.

[22] B. Parno and A. Perrig. Challenges in securing vehicular networks. In *Proc. of Hot Topics in Networks (HotNets)*, November 2005.

[23] T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Proc. of CRYPTO*, pages 129–140, 1991.

[24] M. Raya and J.-P. Hubaux. Securing vehicular ad hoc networks. *Journal of Computer Security*, 15(1):39–68, 2007.

[25] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *Proc. of CRYPTO*, pages 239–252, 1989.

[26] H. Shacham. *New Paradigms in Signature Schemes*. PhD thesis, 2005.

[27] A. Studer, E. Shi, F. Bai, and A. Perrig. TACKing Together Efficient Authentication, Revocation, and Privacy in VANETs. In *Proc. of IEEE SECON*, pages 1–9, 2009.

[28] Vehicle Safety Communications Project Task 3 Final Report Identify Intelligent Vehicle Safety Applications Enabled by DSRC, USDOT National Highway Traffic Safety Assoication, 2005.

[29] VanetMobiSim. `http://vanet.eurecom.fr/`.

Appendix A
Proof of Lemma 1

Proof: We show that if an honest prover has a valid set of credentials A_i, B_i, C_i, and D_i, then the group signature created is always accepted by the verifier. Say the prover computes G, T, F, S, W, K, L, $R_1..R_7$ as in Protocol 2. When the verifier issues a challenge, the prover sends $s_1..s_7$, s_{mult}, s_{tmp}, and s_y. Now the R and $\tilde{R}$ values must match for the signature to be accepted.

For R_1, we use $T = g_2^{t_1} h_2^{t_2}$, $r_1 = s_1 - c \cdot t_1$ and $r_2 = s_2 - c \cdot t_2$:

$$\tilde{R_1} = T^{-c} g_2^{s_1} h_2^{s_2} \equiv g_2^{s_1 - c \cdot t_1} h_2^{s_2 - c \cdot t_2} \equiv g_2^{r_1} h_2^{r_2} = R_1$$

For R_2, we use the fact that $r_{tmp} = s_{tmp} - ct_2t_4$ and $r_{mult} = s_{mult} - ct_1t_4$:

$$\begin{aligned}\tilde{R_2} &= T^{s_4} g_2^{-s_{mult}} h_2^{-s_{tmp}} \equiv g_2^{t_1 s_4 - s_{mult}} h_2^{t_2 s_4 - s_{tmp}} \\ &\equiv g_2^{t_1(r_4+ct_4)-(r_{mult}+ct_1t_4)} h_2^{t_2(r_4+ct_4)-(r_{tmp}+ct_2t_4)} \\ &\equiv g_2^{t_1 r_4 - r_{mult}} h_2^{t_2 r_4 - r_{tmp}} \equiv T^{r_4} g_2^{-r_{mult}} h_2^{-r_{tmp}} = R_2\end{aligned}$$

For R_3, we use $G = B_i h_2^{t_1}$, $F = C_i h_2^{t_3}$, and $e(B_i, f) = e(g_1, C_i)$

$$\begin{aligned}\tilde{R_3} &= \left(\frac{e(G,f)}{e(g_1,F)}\right)^{-c} \cdot e(h_1,f)^{s_1} \cdot e(g_2^{-1}, h_2)^{s_3} \\ &\equiv \left(\frac{e(B_i,f)^{-c} e(h_1,f)^{-ct_1}}{e(g_1,C_i)^{-c} e(g_1,h_2)^{-ct_3}}\right) \cdot e(h_1,f)^{r_1+ct_1} \\ &\quad \cdot e(g_1,h_2)^{-r_3-ct_3} \\ &\equiv \left(\frac{e(B_i,f)}{e(g_1,C_i)}\right)^{-c} \cdot e(h_1,f)^{r_1} \cdot e(g_1,h_2)^{-r_3} \\ &\equiv (1)^{-c} \cdot e(h_1,f)^{r_1} \cdot e(g_1,h_2)^{-r_3} \\ &\equiv e(h_1,f)^{r_1} \cdot e(g_2^{-1},h_2)^{r_3} = R_3\end{aligned}$$

R_4 is more complicated so it will be shown in two steps. First, we will focus on the portion in parenthesis. We use the

fact that $Y = D_i h_2^{t_6}$, $S = A_i h_2^{t_4}$ and $e(pk \cdot B_i, A_i) \cdot e(g_1, D_i) = e(g_1, g_2)$:

$$\begin{aligned}
&\frac{e(pk \cdot G, S) \cdot e(g_1, Y)}{e(g_1, g_2)} \\
&= \frac{e(pk \cdot B_i \cdot h_1^{t_1}, A_i h_2^{t_4}) \cdot e(g_1, D_i \cdot h_2^{t_6})}{e(g_1, g_2)} \\
&= e(pk \cdot B_i, A_i) \cdot e(h_1^{t_1}, A_i) \cdot e(pk \cdot G, h_2^{t_4}) \cdot e(D_i, g_2) \\
&\quad \cdot e(g_1, h_2^{t_6}) \cdot e(g_1, g_2)^{-1} \\
&= e(h_1, A_i)^{t_1} \cdot e(pk \cdot G, h)^{t_4} \cdot e(g_1, h_2)^{t_6}
\end{aligned}$$

Now, for the second part:

$$\begin{aligned}
\tilde{R_4} &= \left(e(h_1, A_i)^{t_1} \cdot e(pk \cdot G, h)^{t_4} \cdot e(g_1, h_2)^{t_6}\right)^{-c} \\
&\quad \cdot e(h_1, S)^{s_1} \cdot e(pk \cdot G, h_2)^{s_4} \cdot e(g_1, h_2)^{s_6} \cdot e(h_1, h_2)^{-s_{mult}} \\
&= e(h_1, A_i)^{s_1 - ct_1} \cdot e(h_1, h)^{s_1 t_4} \cdot e(pk \cdot G, h_2)^{r_4} \\
&\quad \cdot e(g_1, h_2)^{r_6} \cdot e(h_1, h_2)^{-s_{mult}} \\
&= e(h_1, A_i)^{r_1} \cdot e(pk \cdot G, h_2)^{r_4} \cdot e(g_1, h_2)^{r_6} \\
&\quad \cdot e(h_1, h_2)^{r_1 t_4 - r_{mult}} \\
&= e(pk \cdot G, h_2)^{r_4} \cdot e(g_1, h_2)^{r_6} \cdot e(h_1, h_2)^{-r_{mult}} \cdot e(h_1, S)^{r_1} \\
&= R_4
\end{aligned}$$

For R_5 we use $\frac{e(B_i, acc_V)}{e(g_1, w_i)} = z$ and $W = w_i h_2^{t_5}$.

$$\begin{aligned}
\tilde{R_5} &= \left(\frac{e(G, acc_V)}{e(g_1, W) \cdot z}\right)^{-c} \cdot e(h_1, acc_V)^{s_1} \cdot e(g_2^{-1}, h_2)^{s_5} \\
&= \left(\frac{e(B_i, acc_V) \cdot e(h_1^{t_1}, acc_V)}{e(g_1, w_i) \cdot e(g_1, h_2^{t_5}) \cdot z}\right)^{-c} \cdot e(h_1, acc_V)^{s_1} \\
&\quad \cdot e(g_2^{-1}, h_2)^{s_5} \\
&= \left(\frac{e(h_1, acc_V)^{t_1}}{e(g_1, h_2)^{t_5}}\right)^{-c} \cdot e(h_1, acc_V)^{s_1} \cdot e(g_1, h_2)^{-s_5} \\
&= e(h_1, acc_V)^{s_1 - ct_1} \cdot e(g_1, h_2)^{ct_5 - s_5} \\
&= e(h_1, acc_V)^{r_1} \cdot e(g_2^{-1}, h_2)^{r_5} = R_5
\end{aligned}$$

For R_6 we use $Y = D_i h_2^{t_6}$ and $D_i = d_2^{y_i}$:

$$\begin{aligned}
\tilde{R_6} &= Y^{-c} d_2^{s_y} h_2^{s_6} = \left(D_i h_2^{t_6}\right)^{-c} d_2^{s_y} h_2^{s_6} \\
&= d_2^{s_y - cy_i} h_2^{s_6 - ct_6} = d_2^{r_y} h_2^{r_6} = R_6
\end{aligned}$$

R_7 we use $L = \ell^{t_4}$:

$$\tilde{R_7} = L^{-c} \ell^{s_4} = \ell^{s_4 - ct_4} = \ell^{r_4} = R_7$$

■

Appendix B
Proof of Lemma 2

Proof: A simulator for the above knowledge proof may be constructed as follows. First, choose random values for the commitments G, T, F, S, W, Y, L. These are distributed as the real commitments. Then set $c \in_R [1, ..., 2^k]$. Set $s_1, s_2, s_3, s_4, s_5, s_6, s_{mult}, s_{tmp}, s_y \in_R \mathbb{Z}_q^*$ which are distributed as real s values. Now let each of R_1,R_2,R_3,R_4,R_5,R_6,R_7 be set to the value obtained by using the verification equations along with the previous values. For example, let $R_1 \leftarrow T^{-c} g^{s_1} h^{s_2}$ and so on. Because these equations only use the previous values and original values, this transcript is distributed exactly as a real transcript without knowledge of the secrets. ■

Appendix C
Proof of Lemma 3

Proof: We now show the existence of an extractor for the OBU group signature protocol. Suppose that Protocol 2 is run to the point where the challenge is issued. Then, two different challenges c and c' are issued to the prover. The prover responds with s values to both challenges. This is also called 'rewinding' the protocol. Now we have two different values s_1 and $s_1' = r_1 + c' \cdot t_1$. Let $\Delta c \leftarrow c - c'$ and $\Delta s_r \leftarrow s_1 - s_1' = \Delta c \cdot r$. Therefore we have two equalities for $\tilde{R_1}$:

$$\begin{aligned}
\tilde{R_1} &= T^{-c} g_2^{s_1} h_2^{s_2} \mod q \\
\tilde{R_1} &= T^{-c} g_2^{s_1'} h_2^{s_2'} \mod q
\end{aligned}$$

Using the knowledge that $T = g_2^{t_1} h_2^{t_2}$, we divide these equations to get $g_2^{t_1 \cdot \Delta c} h_2^{t_2 \cdot \Delta c} = g_2^{\Delta s_1} h_2^{\Delta s_2}$. Therefore, $t_1 \cdot \Delta c = \Delta s_1$ and $t_2 \cdot \Delta c = \Delta s_2$. Now we have that $t_1 = \frac{\Delta s_1}{\Delta c}$ and $t_2 = \frac{\Delta s_2}{\Delta c}$. Since the secrets are recoverable, Protocol 2 is a convincing knowledge proof. We now show a similar process to extract all the remaining secret values.
Dividing two instances of $\tilde{R_3}$:

$$\begin{aligned}
&\left(\frac{e(G, f_2)}{e(g_1, F)}\right)^{\Delta c} = e(h_1, f_2)^{\Delta s_1} \cdot e(g_1, h_2)^{-\Delta s_3} \\
&\left(\frac{e(B_i, f_2) \cdot e(h_1, f_2)^{t_1}}{e(g_1, C_i) \cdot e(g_1, h_2)^{t_3}}\right)^{\Delta c} = e(h_1, f_2)^{\Delta s_1} \cdot e(g_1, h_2)^{-\Delta s_3} \\
&e(h_1, f_2)^{t_1 \cdot \Delta c} \cdot e(g_1, h_2)^{-t_3 \cdot \Delta c} = e(h_1, f_2)^{\Delta s_1} \cdot e(g_1, h_2)^{-\Delta s_3}
\end{aligned}$$

Therefore we have that $t_1 = \Delta s_1 / \Delta c$ and $t_3 = \Delta s_3 / \Delta c$.
Dividing two instances of $\tilde{R_4}$:

$$\begin{aligned}
&e(h_1, A_i)^{\Delta c \cdot t_1} \cdot e(pk \cdot G, h)^{\Delta c \cdot t_4} \cdot e(g_1, h_2)^{\Delta c \cdot t_6} \\
&= e(h_1, S)^{\Delta s_1} \cdot e(pk \cdot G, h_2)^{\Delta s_4} \cdot e(g_1, h_2)^{\Delta s_6} \\
&\quad \cdot e(h_1, h_2)^{-\Delta s_{mult}} \\
\Rightarrow\ &e(h_1, A_i)^{\Delta c \cdot t_1} \cdot e(pk \cdot B_i, h)^{\Delta c \cdot t_4} \cdot e(pk \cdot h, h)^{\Delta c \cdot t_4 \cdot t_1} \\
&\quad \cdot e(g_1, h_2)^{\Delta c \cdot t_6} \\
&= e(h_1, A_i)^{\Delta s_1} \cdot e(h_1, h_2)^{\Delta s_1 \cdot t_4} \cdot e(pk \cdot B_i, h_2)^{\Delta s_4} \\
&\quad \cdot e(pk \cdot h, h_2)^{\Delta s_4 \cdot t_1} \cdot e(g_1, h_2)^{\Delta s_6} \cdot e(h_1, h_2)^{-\Delta s_{mult}} \\
\Rightarrow\ &e(h_1, A_i)^{\Delta c \cdot t_1} \cdot e(pk \cdot B_i, h)^{\Delta c \cdot t_4} \cdot e(pk \cdot h, h)^{\Delta c \cdot t_4 \cdot t_1} \\
&\quad \cdot e(g_1, h_2)^{\Delta c \cdot t_6} \\
&= e(h_1, A_i)^{\Delta s_1} \cdot e(pk \cdot B_i, h_2)^{\Delta s_4} \\
&\quad \cdot e(pk \cdot h, h_2)^{\Delta s_4 \cdot t_1 + \Delta s_1 \cdot t_4 - \Delta s_{mult}} \cdot e(g_1, h_2)^{\Delta s_6}
\end{aligned}$$

Therefore we have that $t_1 = \Delta s_1 / \Delta c$ and $t_4 = \Delta s_4 / \Delta c$ and $t_6 = \Delta s_6 / \Delta c$ and $t_1 \cdot t_4 = (\Delta s_4 \cdot t_1 + \Delta s_1 \cdot t_4 - \Delta s_{mult}) / \Delta c$.
Dividing two instances of $\tilde{R_5}$:

$$\frac{e(h_1, acc_V)^{t_1 \Delta c}}{e(g_1, h_2)^{t_5 \Delta c}} = \frac{e(h_1, acc_V)^{\Delta s_1}}{e(g_1, h_2)^{\Delta s_5}}$$

Therefore we have that $t_1 = \Delta s_1 / \Delta c$ and $t_5 = \Delta s_5 / \Delta c$.
Dividing two instances of $\tilde{R_6}$:

$$d_2^{y_i \Delta c} h_2^{t_6 \Delta c} = d_2^{\Delta s_y} h_2^{\Delta s_6}$$

Therefore we have that $y_i = \Delta s_y / \Delta c$ and $t_6 = \Delta s_6 / \Delta c$. Now all of $t_1, t_2, t_3, t_4, t_5, t_6, y_i$ have been extracted. The values $mult = t_1 \cdot t_4, tmp = t_2 \cdot t_4$ are derived from the extracted values. Using these values, the commitments S, G, F, W, Y can be derandomized to A_i, B_i, C_i, w_i, y_i, respectively. Therefore, Protocol 2 is a convincing knowledge proof. ■

APPENDIX D
PROOF OF LEMMA 4

Proof: Suppose some algorithm A is able to forge a verifiable signature which opens to the identity of some registered group member. Let A's probability of success be ϵ. We show that an algorithm B is able to break the discrete log assumption in $\mathbb{G}_2$ when given access to algorithm A.

B is given an instance of discrete log problem as $y = g^x$. B starts by carrying out the TREES setup in Section VI-A where $d_2 \leftarrow g$ is specified manually. B then runs the Join procedure and specifies $y_i \leftarrow x$ instead of choosing randomly. A is given all values known to the TA, including d_2 and D_i (but not y_i). In this case, $D_i = d_2^{y_i} = g^x = y$.

In this setting, A can request a hash from H and get a random element in $\mathbb{Z}_q$. This answer will be the same for any future requests of the same parameters. Similarly for H_0. At some point, A outputs a signature σ on message m. Using the extractor protocol of Appendix C, B is able to extract the values $(A_i, B_i, C_i, w_i, y_i)$. Therefore, B has gained advantage ϵ on obtaining $x = y_i$, in contradiction to the discrete log assumption in $\mathbb{G}_2$.

■