Formal Verification of a V2X Privacy Preserving Scheme using Proverif

Simone Bussa, Riccardo Sisto, Fulvio Valenza
Dip. Automatica e Informatica, Politecnico di Torino, Torino, Italy, Emails: {first.last}@polito.it

***Abstract*—V2X communications will be an integral part of all vehicles in the future, broadcasting information such as the vehicle's speed and position to all surrounding neighbors. Being sensitive, a compromise of this data may expose the vehicle to cyberattacks. In this paper, we focus on a particular issue, that is privacy of vehicles and their drivers. Specifically, we consider a scheme that has been proposed in the literature for ensuring privacy in v2x communication, we build a formal model of it and we analyze its security properties through formal verification. Our analysis conducted using Proverif revealed some issues that could impact the privacy and safety of the vehicle. Some of them are well known in the literature and could be common to other existing schemes; other ones are specific of the modeled protocol.**

***Index Terms*—v2x communications, formal verification, vehicle privacy, automotive cybersecurity, proverif**

I. Introduction

The Intelligent Transportation Systems (ITS) area has gained a lot of scientific engagement in recent years, attracting the attention of numerous stakeholders, from governments to major car manufacturers, and it is now finally becoming a road-going reality [1]. In this context, v2x communications could enable the transition from a first generation of purely sensor-assisted vehicles to more complex ones which can communicate with each other, exchange data and discover information beyond their sensory range. The applications that could derive from the use of these technologies are many and both road safety and traffic flow optimizations can benefit from them [2]. To achieve these goals, vehicles must communicate with each other. V2X communications, regardless of the technology with which they are implemented ([3]), allow communication between different vehicles (v2v) or between vehicle and road infrastructure (v2i). They consist of messages transmitted without encryption to all surrounding neighbors (BSM in the USA, CAM and DEMN in EU), containing information such as vehicle position, speed, direction, etc. Since all of this data is sensitive, mechanisms are needed to protect it and prevent a potential attack from improper use.

Traditional network standards typically ensure authentication and integrity of the messages by using asymmetric encryption and a PKI for managing digital certificates. This simple approach is not possible in v2x communications because the use of digital certificates exposes vehicles to privacy issues. An attacker who intercepts all messages associated with the same certificate could link these data as belonging to the same vehicle and trace its position. This is a significant privacy concern. In this sense, vehicle anonymity is a desirable property, but at the same time a certain level of accountability must also be assured, to guarantee a correct service to the user or to discover the real identity of any vehicle that does not behave honestly [4], [5].

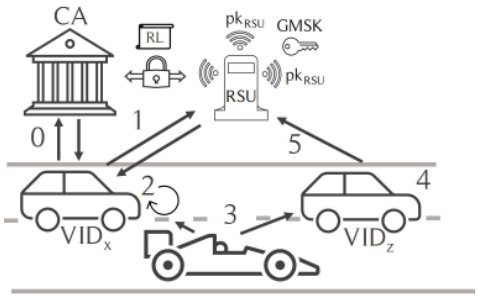
Mixing accountability with anonymity is not trivial. The common way to ensure all these properties is through the use of pseudonyms. A number of schemes have been proposed over the years, to provide vehicles with pseudonyms and allow them to authenticate without revealing their real identity, supported when possible by road infrastructure [6].

Despite this large number of proposed schemes, what is missing in most cases is a formal verification that ensures that the desired properties are actually enforced by the protocols. In this paper, we consider one of the proposed schemes existing in the literature, [7], for which no formal verification has been provided yet. We perform a formal verification using Proverif, which is one of the most commonly used symbolic verification tools [8]. Since the reference paper for some aspects of the scheme just provides a high-level description, without all the necessary steps and details, we propose a way to complete this description and get a formal specification of the protocol. The verification revealed some critical issues that could be common not only to the modeled scheme, but also to others proposed in the literature.

The structure of the paper is as follows. In Section II we present a summary of related work. Section III gives a complete description of the v2x communication scheme that we verified. Section IV shows how this scheme was modeled in Proverif and Section V describes the security properties that the scheme is expected to satisfy. Finally, Section VI presents the verification results and Section VII the conclusions.

II. Related Work

It is well known that designing and implementing network security protocols is an error-prone task. A way to increase the confidence in the final artifact security is through formal verification. Typically, an abstract model of the protocol is built and security properties are defined in an appropriate language with formal semantics. An example of a suitable language for modeling security protocols is the applied pi calculus. After expressing a set of security properties and an abstract model of the system in this language, a formal analysis can be conducted to verify whether these properties are actually enforced by the model. Typically, this analysis



Enrollment certificate: $\{vid_x,\ vpk_x,\ sign_{cask}\}$
Group key request: $\{vid_x, nonce, sign_{vskx}, vcert_x\}_{pkRSU}$
Group key response: $\{vid_x, gpk(gmsk), gsk(vid_x, gmsk),$ $nonce, sign_{cask}\}_{vpkx}$ or $\{vid_x, nonce, error\}_{vpkx}$
Pseudonymous certificates: $\{vpseudopk_j, sign_{gsk}\}$
Message: $\{m, sign_{vpseudoskj}, vpseudocert_j\}$
Revocation report: $\{vpseudcert_{revoke}, sign_{vskz}, vcert_z\}_{pkRSU}$

Fig. 1: Pseudonym credential management scheme

is performed using automatic tools. The tool we used for the automatic verification is Proverif [8].

In the literature, there are numerous works focused on proposing a formal definition of security properties. An example is [9], which shows a model of the most common network security properties. As regards the specific ones connected to privacy instead, some useful works are [10] and [11].

Unfortunately, v2x communications have been subject to little formal verification. And this is a serious lack, since, being a system of distributed nodes that exchange messages through a public network, they lend themselves well to being analyzed with formal verification tools. A paper analyzing a v2x protocol is [12]. The automatic tool used for the verification is Tamarin (i.e., a tool very similar to Proverif). However, this work analyzes only the part of the v2x protocol related to the revocation of a malicious vehicle. Another work of formal verification in the automotive field, this time using Proverif, is [13]. In particular, this paper analyzes an electric vehicle charging protocol. Although it can be considered a sort of v2i protocol, it is very different from the context we want to analyze, with multiple vehicles exchanging messages in a highly dynamic environment. However, we will refer to this paper for the analysis of vehicle privacy-related security properties it made and how they are implemented in Proverif.

III. System model

In this section, we describe the v2x communications scheme that we verified using Proverif. This scheme is taken from [7]. We proposed and added to it some details that were not well specified in the original paper, but which are essential for completing the scheme and building a formal model of it (more on this later). This scheme supports pseudonyms using group signatures and asymmetric encryption.

Group signatures. Vehicles form groups based on their geographic location. A Road Side Unit (RSU) acts as group manager and has a group master secret key, from which it derives the other group keys. By interacting with the RSU, vehicles can get a private group key (unique for the specific vehicle) and a public group key (shared among all vehicles in the group). With the personal private key, each vehicle can make a group signature which is constructed in such a way that any signature made with a private group key can be verified with the same group public key. Moreover, the vehicle that made the signature remains anonymous within the group and it is not possible to link multiple signatures made with the same key together. Only the RSU, which knows the group master secret key, can reveal which private key performed the signature, and consequently extract the vehicle identifier, with an operation called *open*. This system has the advantage that it does not require the generation of pseudonym certificates, to send together with the signed message and to be changed frequently, but only a single private group key, to be updated only in case the vehicle has to change group (i.e., when it moves to a different geographical area). Also revoking a vehicle is very easy, as it is sufficient not to release the group private key to the revoked vehicle (more on this later). The main disadvantage of group signatures is that cryptographic operations are complex and time consuming, so this approach is not feasible if applied to signing (and then verifying) every single message transmitted in broadcast between vehicles.

Asymmetric encryption. To sign messages to be broadcasted, the traditional digital signature using an asymmetric key pair can be used. Standard schemes of this type require the vehicle to request a set of pseudonymous digital certificates from the RSU, usually more than one, to be changed frequently, otherwise an attacker can trace the vehicle through its unique pseudonym. With each message sent, the vehicle attaches the signature (made using the private part of the pseudonym) and the pseudonymous certificate. Asymmetric encryption is much faster than group signatures, so this approach is feasible. The main drawback is the so-called *refill problem*: since pseudonyms have to be changed frequently, a large number of pseudonymous certificates must be requested by the vehicle from the RSU, thus consuming high bandwidth; this is in addition to the fact that the RSU can sometimes be unreachable (due to a lack of connectivity). For these reasons, a large number of pseudonyms must be pre-loaded into the vehicle whenever possible. Furthermore, there are also problems with revocation, since, in order to remove a vehicle, all of its issued pseudonymous certificates must be revoked.

The protocol. The solution modeled in this paper uses pseudonymous certificates with traditional digital signatures for signing messages, with pseudonymous certificates self-certified by the vehicle, without the need to contact the RSU and, thus, avoiding the refill problem. The signature on the pseudonymous certificate is a group signature, done using the vehicle group private key. The reference paper [7] mainly focuses on the basic idea behind this scheme: to use group signatures to self-certify asymmetric certificates. Then it excellently studies the performance of the proposed approach for creating new certificates, self-certifying them, signing messages and verifying both, in terms of time and

overheads. Obtained results are compared to those from other schemes proposed in the literature. However, many aspects of the scheme are not specified in [7]. For example: 1) How do vehicles request group keys and who should they contact to get them? 2) With which credentials and methods do they authenticate to a possible RSU? 3) How do they ask for the revocation of other vehicles? 4) How are the messages composed for the interactions described above? We decided to propose the missing details as follows.

The scheme is shown in Fig. 1, and can be broken down into several steps.

0) Offline registration. Preliminary step in which the vehicle physically goes to the CA and obtains an enrollment certificate, $vcert_x$, with a pair of keys, public, vpk_x, and private, vsk_x, signed by the CA. The enrollment certificate also contains the vid of the vehicle, vid_x; so this certificate cannot be used to exchange messages with other vehicles.

1) Group key provisioning. As soon as the vehicle enters a new geographic region, it contacts the RSU to obtain group keys valid for that region. In the request, the vehicle inserts its vid and a nonce (more on this later), and signs them using its enrollment certificate, which in turn is attached to the request. Everything is encrypted with the RSU public key. The RSU decrypts the message, verifies the validity of the enrollment certificate (i.e., checks if issued by the CA), verifies that the vehicle has not been removed and checks the signature on the request. If all checks are ok, the RSU generates a private group key, gsk_x, computed starting from the vehicle id and the group master secret key, $gmsk$, which is in turn linked to a public group key, gpk. The RSU signs the response and encrypts it with the vehicle public key contained in the enrollment certificate. If something is wrong in the request, the RSU replies with an encryted error message.

2) Pseudonym certificate creation. With a valid group key, the vehicle can generate and self-certify pseudonymous certificates, $vpseudocert_x$, that will be used to sign messages for other vehicles. Therefore, the vehicle generates a key pair, public, $vpseudopk_x$, and private, $vpseudosk_x$, inserts the public one in the certificate, and self-certifies it by signing with the gsk. This process can be repeated multiple times, to generate *n* pseudonymous certificates. These certificates must be changed frequently and have a short life-time.

3) Message sending. To send a message m to nearby vehicles, the vehicle signs it using the $vpseudosk$ linked to the currently active $vpseudocert$ and attaches the $vpseudocert$ to be used for verification. With the same pseudonymous certificate, a vehicle can sign several messages, but in this case they are linkable to each other.

4) Message verification. To verify the authenticity and integrity of the message, the receiving vehicle first verifies the pseudonymous certificate, $vpseudocert$. To do this, the gpk shared among all group members is used: the vehicle cannot discover the specific vehicle that signed the message, but it can only know that the vehicle is a valid member of the group (because it has successfully obtained a group key from the RSU). Then, the receiving vehicle extracts $vpseudopk$ from the certificate, and verifies the signature on the message. Notice that it is not necessary to verify the group signature on the pseudonym certificate every time (because it is a time consuming operation). Once it is ascertained that the $vpseudocert$ is valid, the receiving vehicle can store it on a local list and avoid re-verifying it when it will be received again with the next message. In this case, only the verification of the digital signature on the message is required (and this is a feasible operation).

5) Report and Revocation. When a vehicle detects that it has received a malicious message, it can create a report and notify the RSU to remove the vehicle that sent the message. To do this, it creates a report, inserts the $vpseudocert$ of the vehicle to be revoked, signs it with its enrollment certificate and encrypts everything with the RSU public key. The RSU, in turn, verifies that the report has been done by a valid vehicle and then performs the open operation. Using the $gmsk$, the RSU extracts vid_j from $vpseudocert_{torevoke}$: this is the identifier of the vehicle to revoke. Then, it inserts vid_j in the Revocation List (RL) and updates the $gmsk$. At this point, the RSU notifies all vehicles to request a new group key as soon as possible. The revoked vehicle will fail to obtain the new group key and will no longer be able to participate in the protocol. This is the only part that differs from what is proposed in the reference paper. In [7], the revocation of vehicles is done by distributing a Certificate Revocation List (CRL). Since this is not trivial to model, we preferred another approach suggested in the literature, which is more specific to group signature schemes, [14].

An unbounded number of honest vehicles and malicious vehicles controlled by the attacker can exist simultaneously in the scheme described above. Each vehicle can generate a discretionary number of pseudonymous certificates and use them to sign messages to be broadcasted to other vehicles. Messages are sent on a public network unencrypted and contain the digital signature done using the current active pseudonymous certificate.

IV. Proverif model

In this section, we describe how we modeled the v2x scheme in Proverif [1].

A. Cryptographic operations.

Proverif is an automatic verification tool that uses symbolic models. Unlike computational ones, symbolic models assume perfect cryptography: cryptographic operations are modeled as abstract black boxes, by means of function symbols in an algebra of terms. This means that attacks on the protocol that exploit weaknesses related to specific cryptographic operations are not detected. In our case, we modeled the cryptographic operations of digital signature, asymmetric encryption and group signatures. The equations describing the first two operations are standard, and they can be found in many papers,

[1]The complete source files for Proverif can be found at https://github.com/netgroup-polito/verification-v2x.

as well as in the Proverif manual [8]. The equations for group signatures, on the other hand, have been modeled as follows.

```
(* Group signatures *)
gchecksign(gsign(m, gsk(vid, gmsk)), m, gpk(gmsk)) = ok.
gopen(gsign(m, gsk(vid, gmsk)), gmsk) = vid.
```

The *gchecksign* operation checks the validity of a group signature, *gsign*, on a message *m*, made with a private key, *gsk*, verified using the corresponding group public key, *gpk*. Both *gpk* and *gsk* were constructed from the same *gmsk*, with the private one also tied to the *vid* of the vehicle for which it was created. The *gopen* operation, on the other hand, allows anyone in possession of the *gmsk* to resolve a signature by extracting the vehicle id from the private key used to sign.

B. Attack model.

Proverif follows the Dolev-Yao model. An attacker has full control of messages exchanged on a public network and can delete, modify, replay and forward messages (even when they belong to different sessions). Furthermore, the attacker can read messages if they are sent unencrypted, while if they are encrypted it can read them only if it knows the decryption key. Moreover, it can create new messages from its current knowledge and perform cryptographic operations. Regarding our specific v2x scheme, we have given the attacker the possibility to obtain one or more valid enrollment certificates (correctly signed and issued by the CA), with which it can obtain group keys and participate in the protocol as a legitimate vehicle. The attacker can intercept all messages that are sent between vehicles and vehicles-RSU, as they travel on the public network. On the other hand, it cannot obtain messages sent to/from the CA. This is because vehicles and CA communicate out-of-band, while the communication between RSU and CA is on a protected channel.

C. Protocol phases.

As we described in the previous section, every time a vehicle is revoked, the RSU group master secret key is updated, overwriting the old value with a new one. This introduces a "state" problem. To handle stateful protocols in Proverif, a possible solution is to use phases. All processes can be annotated with a phase prefix, e.g. *phase x*, indicating the specific phase in which the process is allowed to run. A process tied to a particular phase waits until that phase starts. In our case, we modeled a phase transition every time a gmsk update occurs. In particular, the model of the protocol can be split in three distinct phases:

- P0: initial phase. Vehicles register and obtain enrollment certificates and group keys. Then, they exchange messages and participate in the protocol. At some point, a vehicle creates a report and requests the revocation of another vehicle considered suspicious.
- P1: the RSU revokes the vehicle, inserts its id in the RL and updates the group master secret key. Then, it notifies all vehicles to request new group keys.
- P2: vehicles request new group keys and restart the communication.

This was obtained in Proverif as follows. Phase 1 is used for synchronization purposes only. At the beginning, in the main process, two processes are created for each vehicle: one to be started immediately in phase 0 and one waiting in phase 2. The vehicle receives its id and enrollment certificate and can contact the RSU to obtain group keys to communicate with other vehicles. Group keys are released by the *RSUReleaseGroupKey* process, which uses the gmsk to create both the private and public group keys. When the RSU receives a revocation report, the *RSURevoke* process is started. After the various checks, the RSU inserts the vehicle vid in the RL and phase 1 begins. In this phase, the RSU simply updates its group master secret key and launches phase 2, initiating a new *RSUReleaseGroupKey* process with the updated gmsk. Vehicle processes waiting in phase 2 can now run and contact the RSU to obtain updated group keys.

```
process ...
   !( ... create new vehicle ...
      ( Vehicle(vid, vsk, vcert, capk) |
        phase 2;
        Vehicle(vid, vsk, vcert, capk)
      )
   )
   | !RSUReleaseGroupKey(gmsk, ...)
   | !RSURevoke(gmsk, ...)

let RSUReleaseGroupKey(gmsk:gmskey, ...) = ...

let RSURevoke(gmsk, ...) = ...
   insert revocationlist(torevokevid);
   phase 1;
   new updatedgmsk:gmskey;
   phase 2;
   !RSUReleaseGroupKey(updatedgmsk, ...)
```

In this way it is possible to manage the state, but introducing some limitations. The first is that, in doing so, only one vehicle can be revoked. The second concerns the way Proverif handles phases. In particular, during a phase transition, when a process enters a new phase, all others that have not yet reached it are discarded. In our case, when the RSU enters phase 1, all vehicles sending messages in phase 0 are discarded. Despite this, the attacker represents an exception, because it can operate in all phases and carry messages (and other data in its knowledge) from one phase to another. For this reason, thanks to the attacker's ability to "escape" phase constraints, although the verification is not exhaustive, the aforementioned limitations do not heavily affect the obtained results, that still give good guarantees on the validity of the verified properties.

V. Privacy properties

In this section, we describe the security properties that have been verified using Proverif, with a particular focus on those related to privacy. We define these properties and describe how they can be modeled using Proverif.

Sanity Check: simple check that the protocol works as expected (i.e., that all phases of the protocol are reachable). Those checks include, for example, verifying that vehicles, as well as possible adversaries, can obtain group keys, exchange messages, request the revocation of suspicious vehicles etc.

On the contrary, we should check that they cannot do all these operations if they have been revoked. These checks are modeled in Proverif using Events and Correspondence Assertions.

Confidentiality: attackers are not able to obtain secrets of the participants of the protocol, even by actively interacting with them. In our case, the secret to protect is the vehicle's real identity, vid. Confidentiality can be modeled in Proverif as a Reachability property.

$$query\ attacker(vid).$$

Strong secrecy: attackers, unable to discover the secret, cannot even distinguish if it changes. This is useful to capture the attacker's ability to learn partial information about the secret [8]. In our case, we want to verify the Strong secrecy of the vehicle id. With the formula below, we test whether it is possible to replace the vid with different values (e.g., vid_x, vid_y), without the adversary being able to distinguish the two cases. This property in Proverif is modeled as Observational Equivalence, denoted by the keyword *noninterf*.

$$P\{vid_x/vid\} \approx P\{vid_y/vid\}$$
$$noninterf\ vid.$$

Anonymity: a vehicle can participate in the protocol without revealing its identity. In case the identity, for some reason, is known to the attacker, there is anonymity if the vehicle does not reveal that it is using the service. We verified the anonymity of the vehicle when it uses the group secret key and the pseudo secret key to sign respectively pseudo certificates and messages. The formula below tests whether a process P is equivalent to a version of itself in which gsk is replaced by a dummy value. The same could be verified for the $vpseudosk$. This property can be modeled in Proverif as Observational Equivalence using the *choice* construct (equivalent to *noninterf* but to be used with bound names or variables).

$$!(gsk;P) \approx !(gsk;P)|P\{dummygsk/gsk\}$$
$$VehicleSignPseudoCert(choice[vgsk, dummygsk])$$
$$VehicleSignMessage(choice[vpseudosk, dummysk])$$

Unlinkability: a vehicle can participate in the protocol multiple times, without an attacker being able to link them. In our case, until a vehicle changes its pseudonym, all messages signed using the same pseudonymous certificate are linkable to each other. Unlinkability is to be searched in the use of a single group key to sign multiple pseudonymous certificates. The formula below checks whether protocol P, in which the vehicle signs several pseudonymous certificates with the same group secret key, is equivalent to a version of itself in which the number of signatures made is limited to one. In Proverif, unlinkability can be modeled as Observational Equivalence between processes, using the *equivalence* construct.

$$!(P;!sign(gmsk)) \approx !(P;sign(gmsk))$$
$$equivalence$$
$$!(gmsk;VehicleSignPseudoCert(gmsk))$$
$$!(gmsk;!VehicleSignPseudoCert(gmsk))$$

VI. Verification results

In this section we analyze the verification results obtained using Proverif and highlight some weaknesses found by the tool. Some of them are specific to the scheme modeled in this paper, others in common with other schemes.

A. Vehicle continues to be considered valid even after being removed

Between phase 0 (immediately after the RSU has added the revoked vehicle id to the RL) and phase 1 (when the RSU updates the group master secret key), there is a short period of time in which a revoked vehicle can continue to send messages and these would still be considered valid, because the group key hasn't been changed yet. A similar thing can also happen in phase 2. Although the RSU has updated the group master secret key and notified the vehicles about the change, the latter could be "late" and have not yet requested the updated group keys. An attacker, in this period, could continue to use the old group key and vehicles that have not yet updated the group keys would continue to accept it as valid. This problem of Window of Exposure is well known in the literature and also exists in other schemes, for example the ones using Certificate RLs, containing the pseudonymous certificates of all the revoked vehicles (as in [7]). As long as the vehicles do not download the updated CRL, the Window is open and attackers can continue to operate.

A solution proposed in the literature and adopted in [7] and [14] as well is obtained by releasing group keys with short life. In this way, every certain time, the vehicles must update their keys, both public and private, even if there have been no revocations. This reduces the Window of Exposure, with the drawback of requiring more key updates and, thus, more interaction with the RSU.

B. Pseudonymous certificate continues to be used even after the vehicle has been removed

An attack trace returned by Proverif shows this weakness, which occurs when there are multiple attacker vehicles participating in the communication. An attacker vehicle is removed, but its pseudonymous certificate which signed malicious messages can be taken by a second not-revoked attacker, which managed to obtain an updated group private key, which can re-certify it using that key. The pseudonymous certificate that has been reported continues to be usable thanks to the updated signature done by the second attacker. This process can continue as long as there is a not-revoked attacker in the group, but it cannot last forever: since removed vehicles cannot get new group keys, once all attackers are removed from the group, they are excluded from the protocol.

A similar problem exists when the attacker shares its private group key with other attacking vehicles. This was reported by Proverif as well. In the modeled scheme, a private group key is unique for each vehicle: it is built from the vehicle id and the gmsk, therefore, even if an attacker requests multiple group keys at different times, as long as the RSU does not change the gmsk, the private key it receives is always the

same. Thanks to this, the RSU does not have to check, when issuing a key to a vehicle, whether the vehicle has already obtained that key in the past. Therefore, an attacker sending multiple requests with the same id cannot obtain multiple private keys to distribute to other attackers. However, it can just share its unique group private key. Group signatures, due to the way they are constructed, do not allow vehicles to discover the group private key used for a signature or to link different signatures made with the same key. Therefore, several messages received from n different vehicles could be signed with the same group secret key, which should belong to a single vehicle, without the receiving vehicle realizing it. This is a problem intrinsic to the concept of anonymity, and it is shared with the other schemes. For example, for schemes in which pseudonymous certificates are issued by the CA, the attacker is not prevented from requesting several pseudonymous certificates and then distributing them to other attacking vehicles. In the literature, several studies are trying to solve this problem. A possible solution proposed by [15] is to equip each vehicle with an HSM. Group keys are directly "installed" by the RSU into the HSM, using a secure channel (e.g., encrypting messages with asymmetric keys). The HSM then performs the various operations without releasing the keys to the vehicle. The attacker would never know its keys and certificates, and could not share them with other attackers.

C. Anonymity issue in the revocation phase

We looked at what would happen if the RSU, when receiving a key update request from a revoked vehicle, simply ignored it without sending a response. Proverif discovered this attack which could violate the anonymity of any removed vehicle, with an attacker able to understand which specific vehicle has been revoked. The attacker intercepts and stores an initial request sent from a vehicle to the RSU to obtain a group key. A vehicle is then revoked. The attacker replying the stored request could understand if the revoked vehicle is the one it has initially intercepted, based on whether it can get a response from the RSU. This attack is only effective if the attacker is initially able to link the intercepted request to the specific vehicle that sent it (for example, if they were initially the only vehicles on a traffic-free road). From that moment on, the anonymity of the vehicle in the revocation phase could be compromised. To avoid this problem it is sufficient that the RSU sends a response message to key update requests even if the requesting vehicle is a revoked one. Being the response encrypted with the vehicle's public key, the attacker would not be able to distinguish whether it contains a correct group key or an error message.

D. Linkability issue in the request for a group key

The nonce in the group key request is necessary to avoid a possible Linkability attack. Requests for a group key in the modeled scheme, aside from the nonce, contain vehicle id, signature made using the enrollment certificate and the enrollment certificate as well, all encrypted with the RSU public key. These are all constant data that do not change over time. Therefore, two different requests, sent at different times from the same vehicle, are identical and can be linked to each other. Inserting a nonce in the request would solve the problem, since the message, when encrypted, would always result in different data.

VII. Conclusions

In this paper we formally modeled and analyzed a scheme proposed in the literature that can be used to enforce privacy preserving security properties in v2x communications. The formal analysis was done using Proverif. The obtained results showed that the version of the protocol satisfies the expected security properties, with the exception of some issues well known in the literature, for which scientific research is currently working to propose a solution. These weaknesses are, in fact, common also to other schemes.

As future work, the model can be extended by formalizing and verifying new security properties, as well as contributing to solve the known issues and proposing a robust and correct implementation of the protocol. In addition to this, the work of formal verification done in this paper can be repeated on other existing schemes proposed in the literature, to check if they suffer from the same issues or to find new undiscovered vulnerabilities.

References

[1] K. N. Qureshi and A. H. Abdullah, "A survey on intelligent transportation systems," *Middle-East J. of Scientific Research*, vol. 15, no. 5, 2013.

[2] Z. MacHardy, A. Khan, K. Obana, and S. Iwashina, "V2x access technologies: Regulation, research, and remaining challenges," *IEEE Commun. Surveys & Tuts.*, vol. 20, no. 3, 2018.

[3] S. Al-Sultan, M. M. Al-Doori, A. H. Al-Bayatti, and H. Zedan, "A comprehensive survey on vehicular ad hoc network," *J. of Net. and Comp. Applications*, vol. 37, 2014.

[4] F. Schaub, Z. Ma, and F. Kargl, "Privacy requirements in vehicular communication systems," in *In the Proc. of the IEEE Inter. Conf. on Computational Science and Engineering*, vol. 3, 2009.

[5] D. J. Glancy, "Privacy in autonomous vehicles," *Santa Clara L. Rev.*, vol. 52, p. 1171, 2012.

[6] J. Petit, F. Schaub, M. Feiri, and F. Kargl, "Pseudonym schemes in vehicular networks: A survey," *IEEE Commun. Surveys & Tuts.*, vol. 17, no. 1, 2014.

[7] G. Calandriello, P. Papadimitratos, J.-P. Hubaux, and A. Lioy, "Efficient and robust pseudonymous authentication in vanet," in *In the Proc. of the ACM Inter. Work. on Vehicular ad hoc networks*, 2007.

[8] B. Blanchet and B. Smyth, "Proverif 1.85: Automatic cryptographic protocol verifier, user manual and tutorial," 04 2011.

[9] M. D. Ryan and B. Smyth, "Applied pi calculus," in *Formal Models and Techniques for Analyzing Security Protocols*. Ios Press, 2011.

[10] M. Brusó, K. Chatzikokolakis, S. Etalle, and J. Den Hartog, "Linking unlinkability," in *In the Proc. of the Inter. Symp. on Trustworthy Global Computing*, 2013.

[11] M. Arapinis, T. Chothia, E. Ritter, and M. Ryan, "Analysing unlinkability and anonymity using the applied pi calculus," in *In the Proc. of the IEEE computer security foundations symposium*, 2010.

[12] J. Whitefield, L. Chen, F. Kargl, A. Paverd, S. Schneider, H. Treharne, and S. Wesemeyer, "Formal analysis of v2x revocation protocols," in *In the Proc. of the Inter. Work. Security and Trust Management*, 2017.

[13] M. Fazouane, H. Kopp, R. W. van der Heijden, D. Le Métayer, and F. Kargl, "Formal verification of privacy properties in electric vehicle charging," in *In the Proc. of the Inter. Symp. of Engineering Secure Software and Systems*, 2015.

[14] X. Lin, X. Sun, P.-H. Ho, and X. Shen, "Gsis: A secure and privacy-preserving protocol for vehicular communications," *IEEE Transactions on vehicular technology*, vol. 56, no. 6, pp. 3442–3456, 2007.

[15] Sevecom project. [Online]. Available: https://www.sevecom.eu/