# Accountable De-anonymization in V2X Communication

Aku Silvennoinen

Master's thesis
UNIVERSITY OF HELSINKI
Department of Computer Science

Helsinki, December 12, 2017

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

| Tiedekunta — Fakultet — Faculty | Laitos — Institution — Department |
|---|---|
| Faculty of Science | Department of Computer Science |

Tekijä — Författare — Author
Aku Silvennoinen

Työn nimi — Arbetets titel — Title

Accountable De-anonymization in V2X Communication

Oppiaine — Läroämne — Subject
Computer Science

| Työn laji — Arbetets art — Level | Aika — Datum — Month and year | Sivumäärä — Sidoantal — Number of pages |
|---|---|---|
| Master's thesis | December 12, 2017 | 71 |

Tiivistelmä — Referat — Abstract

De-anonymization is an important requirement in real-world Vehicle-to-Everything (V2X) systems (e.g., to enable effective law enforcement). In de-anonymization, a pseudonymous identity is linked to a long-term identity in a process known as *pseudonym resolution*. For de-anonymization to be *acceptable* from political, social and legislative points of view, it has to be *accountable*. A system is accountable if no action by it or using it can be taken without some entity being responsible for the action. Being responsible for an action means that the responsible entity cannot deny its responsibility of or relation to an action afterwards. The main research question is: *How can we achieve accountable pseudonym resolution in V2X communication systems?* One possible answer is to develop an accountable de-anonymization service, which is compatible with existing V2X pseudonym schemes. The accountability can be achieved by making some entities accountable for the de-anonymization. This thesis proposes a system design that enables, i) fine-grained pseudonym resolution; ii) the possibility to inform the subject of the resolution after a suitable time delay; and iii) the possibility for the public to audit the aggregate number of pseudonym resolutions. A Trusted Execution Environment (TEE) is used to ensure these accountability properties. The security properties of this design are verified using symbolic protocol analysis.

ACM Computing Classification System (CCS):
• **Security and privacy ~ Privacy-preserving protocols** • **Security and privacy ~ Privacy protections** • *Security and privacy ~ Formal methods and theory of security* • *Security and privacy ~ Security in hardware* • *Security and privacy ~ Social aspects of security and privacy*

Avainsanat — Nyckelord — Keywords
V2X, ITS, de-anonymization, accountable pseudonym resolution, TEE

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

# Contents

# Abbreviations

**APRS** accountable pseudonym resolution service

**CID** checking ID

**CIDS** checking ID storage

**CoPRA** Conditional Pseudonym Resolution Algorithm in VANETs

**CPU** central processing unit

**DPA** data protection agency

**IBC** identity-based cryptography

**ID** identifier

**I.R.A.G** Internal, Rational, Active and Global

**LA** linkage authority

**LTCA** long term certificate authority

**PCA** pseudonym certificate authority

**PFM** public forum message

**PIA** pseudonym issuing authority

**PKI** public key infrastructure

**PUCA** A pseudonym scheme with strong privacy guarantees for vehicular ad-hoc networks

**RA** registration authority

**REE** rich execution environment

**SCMS** A Security Credential Management System for V2V Communications

**SGX** software guard extensions

**TEE** trusted execution environment

**V2I** Vehicle-to-Infrastructure

**V2V** Vehicle-to-Vehicle

**V2X** Vehicle-to-Everything

# 1 Introduction

Communication is an essential part of intelligent transportation systems. The term intelligent transportation system comprises all modes of transportation [17]. In intelligent transportation systems information and communication technologies are utilized to achieve for instance better road safety. intelligent transportation system communication is often referred to V2X communication. In vehicular transport, vehicles are identifiable and trackable because of license plates [15], but V2X communication should not make security or privacy worse. Security and privacy are important in V2X communication. Security is important because V2X communication is used for road safety applications and privacy is important because for instance uncontrolled tracking of vehicles makes it possible to predict movements of individuals.

*Identity management systems* for V2X have been the subject of extensive research. The first approaches to ensure privacy were based on asymmetric cryptography and *digital pseudonyms*. These approaches have been utilised in V2X development projects and standardisation, such as SeVeCom [29], [26], the standard IEEE 1609.2-2016 [3], and ETSI intelligent transportation system standards [1], [2] and [4]. Identity management can be done by using a Public Key Infrastructure (PKI) based V2X pseudonym scheme [31]. In such schemes, concealing the vehicle's (user's) identity from other users and non-authorised entities is among the highest priorities. However, in some situations the possibility to link a pseudonym to its holder is desirable. For instance, if a vehicle escapes from an accident scene trying to hide its identity. The linking is called *pseudonym resolution*. The term pseudonym resolution is a synonym for the therm de-anonymization. The term de-anonymization is understandable for a wide audience, but it is not suitable for treating connections between long-term identities and pseudonyms in a more granular way. Therefore the term pseudonym resolution is used in this thesis.

Pseudonym resolution schemes exist (e.g. CoPRA [8]). Pseudonym resolution schemes can also be called *identity escrow* schemes. For a pseudonym resolution scheme to be *acceptable* from political, social and legislative points of view, it has to be *accountable*. In this thesis, *the term accountability refers to accountability of pseudonym resolution*. In V2X, vehicles can be accountable. In this thesis that accountability is called *accountability of vehicles*. An example of accountability is the so-called *obligation to inform* in Swiss law, which states that the target of resolution should, possibly after some time, know that its identity had or has been resolved [27]. Another example of accountability is to publish statistical information about identity resolution activities (e.g. how many vehicle identities have been resolved etc.).

## Research Goals

The main research goal of this thesis is to answer the question:

*How can we achieve accountable pseudonym resolution in V2X communication systems?*

Answering this question must consider the security and privacy requirements in V2X communication. The approach of this thesis is to answer this question by defining a set of requirements for accountable pseudonym resolution, proposing a design for such a system, and evaluating the design in terms of the defined requirements.

## Solution Overview

The solution is a design of a generic pseudonym resolution system which is modular by design so that it is compatible with many pseudonym schemes. The solution is presented in Section 5. The solution introduces an accountable entity called a judge, which authorizes a law enforcement authority to for example resolve a pseudonym by giving a resolution order. Whenever a pseudonym resolution happens, data of it is published. Additionally, the judge can decide to inform targets of pseudonym resolution by sending an *order to inform*. When the system receives the order, it publishes the target data and additional metadata of all the pseudonym resolutions which are done with the related pseudonym resolution order. First, the relevant background information, including an overview of pseudonym schemes, is presented in Section 2. Next, the problem definition and an adversary model are described in Section 3. Section 4 presents a set of requirements for accountable pseudonym resolution. Section 5 describes the proposed solution in detail. The security evaluation for the design is presented in Section 7. Section 8 discusses related work, and Section 9 presents the final conclusions.

# 2 Background

## 2.1 Intelligent Transportation Systems

Vehicular ad hoc networks are developed from mobile ad hoc networks to enable ad hoc communication in intelligent transportation systems. vehicular ad hoc network consists of Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications which are supported by for instance IEEE 802.11p [18]. V2X denotes both V2V and V2I. The participating entities in V2X are vehicles and infrastructure elements, such as road side units. The vehicle communicates with a system through an on-board unit.

## 2.2 V2X Pseudonym Schemes

Realizations of V2X pseudonym schemes are a subset of identity management systems for V2X. The term identity management is discussed and defined by Pfitzmann & Hansen [32, page 33]. In this thesis, the term *identity management system* denotes a V2X identity management system which utilizes pseudonyms. Since one of the goals is to present a general pseudonym resolution service for identity management systems, it is of great importance to treat pseudonym schemes on the abstract and general level in order to design a compatible service. In addition, examples of pseudonym schemes are discussed.

For a vehicle to participate to an identity management system, it should be enrolled to it. In vehicle enrollment, a long-term identity is given for it. Long-term identity is used for the vehicle to authenticate itself to the identity management system.

The most important purpose of a V2X pseudonym scheme is to preserve the anonymity of its users – vehicles. Anonymity is achieved by issuing many pseudonyms for a vehicle so that it can periodically change the pseudonym it uses to authenticate itself as a legitimate participant of an identity management system. A pseudonym can be some cryptographic key material, and in many schemes, traditional PKI certificates are used in associating these keys to authorization data.

Thwarting tracking is also important. For instance, without tracking thwarting in V2X, it could be possible for *everyone* to do mass surveillance with relatively low resources by setting up a surveillance station network. Tracking thwarting is a difficult problem because of for instance license plates [15]. But if the examination is limited to only digital V2X messages, messages to and from on-board units, tracking can be thwarted effectively by using pseudonyms.

A realistic pseudonym scheme contains more than just pseudonym issuing. Pseudonym revocation denotes that issued pseudonyms are invalidated. Vehicle revocation denotes that a vehicle is invalidated within the system – it cannot get valid pseudonyms anymore.

In a pseudonym scheme, resolution information can be saved for pseudonym resolution. Resolution information can be called also *escrow data* if resolution information is like an escrow – "a bond, deed, or other document kept in the custody of a third party and taking effect only when a specified condition has been fulfilled" [41].

### 2.2.1 Security Requirements

Vehicular communication should be carefully secured to maintain road safety. Pseudonym schemes can be used as a part of preserving security in vehicular communication.

**Authentication:** In vehicular communication, participants should be able to trust received messages – they should be able to verify the authenticity of received messages. Message authentication contains sender authentication and message integrity. Sender authenticity verification comprises at least checking that the sender is a legitimate member of the system. This requirements is discussed also in [33] and [36].

**Accountability of vehicles:** Accountability of vehicles is a natural requirement in vehicular communication since the environment is safety-critical. For instance, forged warning messages should not exist without consequences. For holding a perpetrator accountable, law enforcement authority should be able to identify the perpetrator. Therefore for assigning liability, accountability of vehicle is required. Accountability of vehicle covers non-repudiation. In the V2X context non-repudiation means that a vehicle cannot deny that it sent a message. This accountability of vehicles requirement is discussed also in [33] (by using the term non-repudiation) and [36].

**Restricted credential usage:** Authentication credentials should be restricted in time. This means that a credential is valid for limited time. With time-restricted credentials, it is possible to control how many credentials a vehicle has. Parallel use of pseudonyms should also be restricted. If parallel use is not restricted, then a vehicle could mimic many vehicles at the same time and that can lead to the so called Sybil attacks. The Sybil attack refers to forging and using identities in peer-to-peer systems [14]. This requirement is discussed also in [43] and [36].

**Credential revocation:** Credentials should be revocable. Revoking all credentials of a vehicle results in excluding the vehicle from the system. It should be possible to isolate a malfunctioning vehicle from the system. The subject of the credential revocation can be ephemeral (pseudonyms in a pseudonym scheme) or long-term identity credentials. This requirement is discussed also in [21] and [36].

### 2.2.2 Privacy Requirements

Privacy is critical in vehicular communication – for instance, vehicle tracking should be thwarted as effectively as possible.

**Unlinkability:** Relations between items of interest should not be revealed. Items of interest can be for example subjects, messages, and actions. Items of interest in pseudonym schemes are long-term identities and pseudonyms. Unlinkability of pseudonyms is important to thwart tracking (V2X communication based) effectively. This requirement is discussed also in [32] and [36].

The opposite of the unlinkability is the *linkability*. Linkability means the ability of finding out relations between items of interest.

**Anonymity:** Given a message, the sender of the message should be anonymous within the *anonymity set* – the set of potential message senders[1]. This requirement is discussed also in [32], [31], and [36].

**Trusted or distributed resolution authority:** In identity resolution, the capability to resolve should be allowed only to trusted authorities *or* capability to resolve should be distributed – no single authority should be capable of resolving identities alone. This requirement is partially discussed in [8] (distributed resolution authority) and [36] (distributed resolution authority).

**Perfect forward privacy:** If one credential is resolved to an identity, then the resolution should not reveal any information that can weaken unlinkability of other credentials. This requirement is discussed also in [32] and [36]

The authors of [36] mention the requirement of minimum disclosure which states that revealed information of a communication participant should be minimal. It is a good guideline for protocol designers and implementers.

### 2.2.3 Requirement Relations

Anonymity and accountability of vehicles requirements are in obvious conflict. Thus anonymity and accountability of vehicles should be balanced in a way that the compromise satisfies at least users (vehicles) and law enforcement authority. Unlinkability of a long-term identity and a pseudonym is closely related to the anonymity requirement.

### 2.2.4 Abstract Pseudonym Life Cycle

In [31], the authors present an abstract pseudonym life cycle for vehicular networks. The life cycle can be found in figure 1.

[1] The set of potential message senders can be for example all the vehicles in the system.

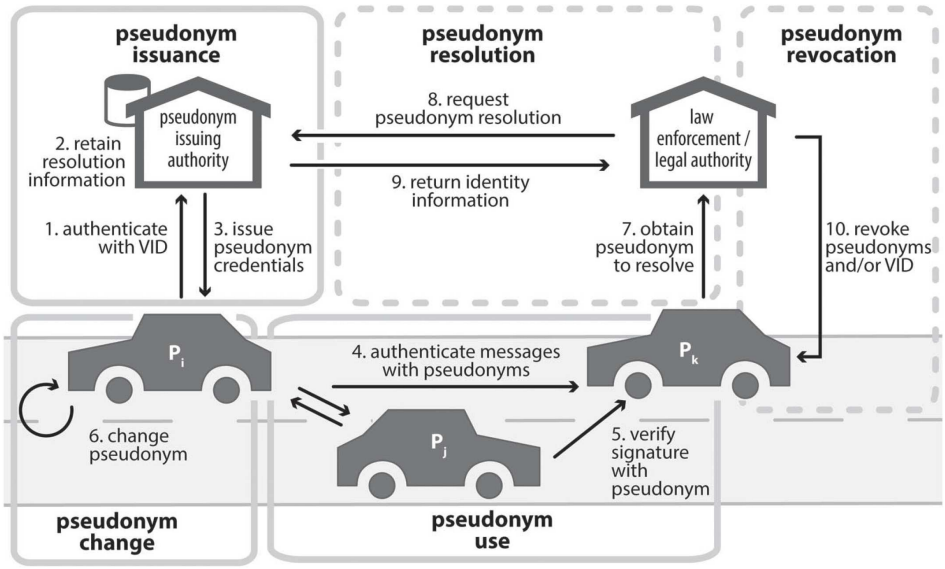


Figure 1: Abstract pseudonym life cycle [31] © 2014 IEEE

**Pseudonym issuance** Vehicle $P_i$ requests pseudonyms from the Pseudonym Issuing Authority (PIA). If pseudonyms are issued by a PIA which is distinct from a vehicle, then this is called *third-party issuance*. In turn, if a vehicle doesn't request pseudonyms from a distinct entity, then this is called self-issuance. When a PIA is issuing requested pseudonyms, it retains resolution information, though in some schemes resolution information is not saved [21]. Finally, issued pseudonyms are delivered to the vehicle. A pseudonym has a validity time. That is why there is an interval for a vehicle to request a new set of pseudonyms.

**Pseudonym change** Vehicle $P_i$ changes its pseudonym in order to prevent tracking and revealing its real identity. In other words, the vehicle prevents the linkability of its actions by performing distinct actions under distinct pseudonyms. If all pseudonyms are expired, then new pseudonyms should be requested.

**Pseudonym use** A vehicle sends a message to another vehicle or infrastructure node. The message is authenticated with a pseudonym by proving that the sender has valid credentials to the system, the pseudonym. Specifically, authentication can be done for instance by signing the message with a private key related to the pseudonym or calculating a message authentication code of the message. The receiver of the message can check that the sender is a legitimate member of the system by verifying the message. Verification can be done e.g. by checking a signature with a pseudonym certificate or calculating a message authentication code. In some schemes, checking can be done instantly, for instance in [43] if corresponding Pseudonym Certifi-

cate Authority (PCA) certificate is available instantly. In other schemes, for instance in [12], there is a inherent verification delay because message authentication codes are verified in base stations.

**Pseudonym revocation** In many schemes, pseudonyms can be revoked. If a pseudonym is revoked, the pseudonym cannot be used in authentication anymore. Revocation can be used if a vehicle is misbehaving and a law enforcement authority entity detects it. When detected, law enforcement authority can ask a scheme specific authority to revoke a pseudonym or pseudonyms. More realistic revocation is vehicle ID revocation. In vehicle ID revocation a vehicle cannot get any more pseudonyms.

**Pseudonym resolution** Pseudonym resolution refers to linking a pseudonym to the corresponding long-term identity or linking two or more pseudonyms of the same holder together. vehicle ID revocation is dependent on pseudonym resolution. Pseudonym resolution can be requested by an law enforcement authority. A request is handled by a PIA or similar authority.

### 2.2.5 Pseudonym Scheme Classification

Pseudonym schemes can be divided into four classes based on the used cryptography [31]. In this section, the characteristics of each class are presented. At least pseudonym issuance and pseudonym resolution are treated for each class since these aspects are the most relevant for pseudonym resolution.

#### Asymmetric Cryptography

The class covers all the schemes where asymmetric cryptography is used. Key pairs of private and public keys are used in asymmetric cryptography. An example of this is the SCMS system (described in section 2.2.6). In the class, it is natural to use traditional PKI. For each key pair, there is a certificate which contains at least the public key and a signature of the public key. The signature is from a PIA for proving that the key pair holder is a legitimate participant of a system. In pseudonymous certificates, there is no data identifying a sender. Messages are signed with pseudonymous or long-term private keys and corresponding certificates are sent within the messages.

**Pseudonym issuance** in this class is similar to certificate issuance in a PKI. Pseudonymous certificates are issued by a pseudonym provider or similar authority.

**Pseudonym use** is similar to certificate use in a PKI – when a vehicle sends a V2X message, it chooses a pseudonymous certificate, signs the message with the private key for the certificate and sends the message with the signature and the certificate. The receiver checks the signature with the certificate and ensures that the sender is a valid member of the system just like in a PKI.

**Pseudonym resolution** can be performed in some schemes by cooperation of pseudonym Registration Authorities (RAs), for instance [43] (section 2.2.6). RAs have the resolution data or RAs are authorized to get the data from pseudonym providers, certificate authorities or other authorities [29]. There exist pseudonym schemes where linking between vehicle's identity and a pseudonym cannot be resolved without vehicle's cooperation [21]. In such schemes, vehicles should *provide* their identities for the resolution process.

*Self-issuing* is possible in asymmetric schemes. Armknecht *et al.* [5] utilizes PKI+ [45] in a self-issuing pseudonym scheme. In the scheme, pseudonym resolution is possible without help from a vehicle.

### Identity Based Cryptography

IBC [38] is similar to traditional public key cryptography. In IBC, there are also private and public keys. But in IBC, a public key is an identity, which can be just a string, and a private key is derived from a public key. Private keys are generated by a trusted central authority and given to vehicles. To generate a private key for a public key in IBC all system parameters should be known. IBC can be used in a pseudonym scheme so that all parameters are not revealed to vehicles.

**In pseudonym issuance**, pseudonymous identities can be fresh strings. In IBC-based schemes, the pseudonym provider is often called the trusted authority. In addition to delivering private keys for pseudonymous identifiers, the trusted authority delivers also parameters for signature verification.

**In pseudonym use** vehicles can sign messages with their private keys and signatures can be verified with public keys which are pseudonymous.

**Pseudonym resolution** can be done by the trusted authority if mappings from long-term identities to pseudonymous identities are saved when issuing pseudonyms.

In IBC there is no PKI nor certificates. Within an IBC system, signed messages are just checked with identities delivered with messages. A signature can be trusted because only a centralized trusted authority knows all the parameters and is thereby able to deliver private keys. An IBC-based scheme can be found for instance from [25].

### Group Signature Cryptography

In a group signature scheme [11], a group member can sign a message on behalf of the group. In the V2X context, vehicles can form a group and if a vehicle of the group signs a message with the group key a receiver can be sure the sender belongs to the group but it cannot infer that *which* member the sender is. In many group signature schemes, there is a group manager

which is responsible for adding members and generating group parameters, for instance, [24].

**Pseudonym issuance** in a group signature pseudonym scheme can be interpreted to be public key delivery to a new member or delivery of a new public key to each member in case of group parameters changing. Each member has its own private key, which can also be delivered in member addition.

**In pseudonym use** a sender signs a message with its private key. A receiver can check the signature with group public key.

**Pseudonym resolution** in a group signature pseudonym scheme is called signature anonymity revoking. In signature anonymity revoking, the group manager maps a signature to a long-term identity.

In a group signature pseudonym scheme, the group manager can be a vehicle or it can be some trusted infrastructure entity.

#### Symmetric Cryptography

Choi, Jakobsson, and Wetzel [12] introduced one of the first plausible symmetric pseudonym schemes. In the scheme, an entity called the ombudsman generates a unique identifier and a seed value for a vehicle in the registration phase. ombudsman and the vehicle can then calculate a set of pseudonymous handles by hashing the unique identifier, the seed value, and a counter value. ombudsman retains escrow data in registration.

**Pseudonym issuance** happens in cooperation between a vehicle and e.g. an road side unit. A vehicle sends one of its handles to an road side unit to request new pseudonyms. The RSU generates pseudonyms by hashing the handle with time values. Pseudonyms are keys for a symmetric cipher. IDs are given for generated pseudonyms. RSU retains escrow data.

**In pseudonym use** signing can be done by generating a hashed message authentication code of a message. The Identifier (ID) of the used pseudonymous key must be provided with the signature. A receiver should send a received message to a road side unit for verification, since the receiver does not know the key for the ID.

**Pseudonym resolution** can be done by an RSU and the ombudsman.

### 2.2.6 A Security Credential Management System for V2V Communications

SCMS [43] is a scheme that uses asymmetric cryptography and traditional PKI. In SCMS, most of the requirements are achieved by organisational separation. For instance, pseudonym resolution data is saved in issuing process, but in such a way that more than one authority is needed for resolution. Trust among participants is delivered with PKI and traditional certificates. Participating entities are shown in figure 2. In the legend of the figure 2, In-

trinsically Central denotes that a component can have "exactly one distinct instance for proper functioning" [43]. Whereas Not Intrinsically Central denotes that there can be $n$ instances of a component. As can be seen, the system has many entities because of organizational separation. Before pseudonyms can be issued to a vehicle, the vehicle has to be enrolled via the Enrollment certificate authority and Device Configuration Manager in the figure 2. In enrolment, a vehicle gets an enrollment certificate which can be used to request pseudonyms. This section explains i) vehicle enrollment ii) pseudonym issuance, and iii) pseudonym resolution in SCMS.

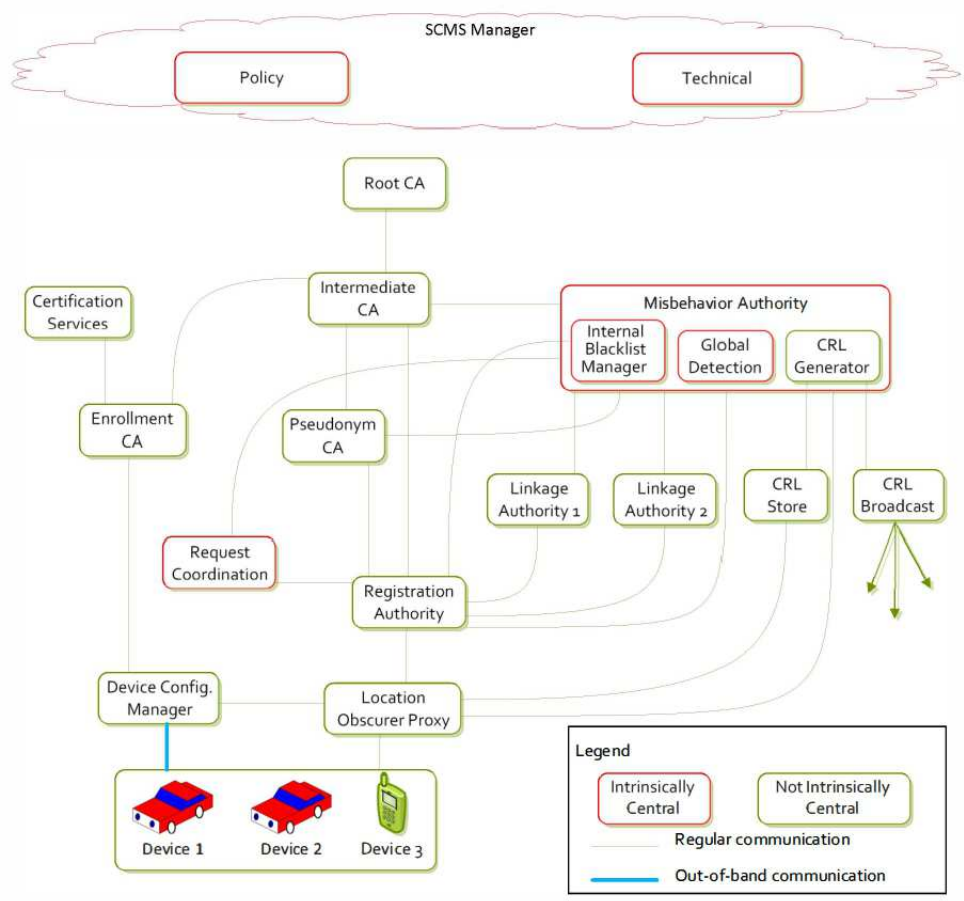


Figure 2: SCMS overall system architecture [43] © 2013 IEEE

**Butterfly key expansion:** SCMS uses a novel technique called butterfly key expansion in pseudonym issuance. It is a cryptographic scheme where it is possible to request arbitrarily many certificates with different key pairs. Each certificate and key pair can be encrypted with a different encryption key. In the request, only one verification public key seed, one encryption public key seed and two expansion functions are needed. With

this technique it is possible to avoid a vehicle to generate and send a distinct encryption key for each pseudonym. The main points are explained in this section, and interested readers are referred to the paper for further details. Elliptic curve cryptography (ECC) and the discrete logarithm problem are used in this explanation. First, all parties agree on a base point $G$. It has order $l$. A *caterpillar keypair* $(\alpha, \alpha G)$ is formed using $G$ and an integer $\alpha$, and $\alpha G$ is denoted as $A$. The requester defines an expansion function $f_k(\iota)$, which is a pseudo-random permutation ([22]) from integers mod $l$ to integers mod $l$ and delivers it to a receiver, along with $A$. The key $k$ defines the function $f$. If an adversary knows tuple of values and corresponding arguments of a pseudo-random function, the adversary cannot derive any other value-argument pairs without knowing the $k$. The receiver of the request can then generate *cocoon public keys* $B_\iota = A + f_k(\iota) * G$. In practice, the receiver calculates keys by substituting integers mod $l$ to the variable $\iota$ one by one. Each of the results are in the same set integers mod $l$. Corresponding private key for each cocoon public key $B_\iota$ is $b_\iota = \alpha + f_k(\iota)$, because

$$B_\iota = A + f_k(\iota) * G = \alpha G + f_k(\iota) * G = (\alpha + f_k(\iota)) * G = b_\iota G.$$

Note that $\alpha$ cannot be derived from $A$, because of the discrete logarithm problem. Therefore public keys are known by the receiver, but the corresponding private keys are known only by the requester. The final step of the butterfly key expansion is to generate a random integer $c$ and obtain $C = cG$ to get a *butterfly public key* $B_\iota + C$. Because

$$B_\iota + C = A + f_k(\iota) * G + cG = \alpha G + f_k(\iota) * G + cG = (\alpha + f_k(\iota) + c) * G,$$

the private key for $B_\iota + C$ is $\alpha + f_k(\iota) + c$.

### Pseudonym Issuance

**Pseudonym request:** A vehicle sends a pseudonymous certificate set request which is signed with an enrollment certificate to location obscurer proxy. In the request, the vehicle sends also public butterfly key seeds $(A, f_k)$ and $(H, f_e)$, where $A$ and $H$ are second elements of caterpillar key pairs and $f_k$ and $f_e$ are corresponding expansion functions with keys $k$ and $e$. The request is encrypted for a RA. The location obscurer proxy obscures origin data of the message and sends it to the RA. When the RA receives the message, it decrypts it and verifies the enrollment certificate. It proceeds with checking that is the vehicle allowed to get new pseudonyms. If all checks succeed, then the RA sends an acknowledgement to the vehicle and proceeds with butterfly key expansions $(A, f_k)$ and $(H, f_e)$. The tuple of keys from the butterfly key expansion $(A, f_k)$ is denoted by $B^A$ and the $\iota$-th key in the tuple is denoted by $B^A_\iota$. The tuple of keys from the $(H, f_e)$ similarly.

**Request batching:** The RA waits till it has enough pseudonymous certificate requests and pre-linkage values from each of the linkage authorities. Linkage authorities generate pre-linkage values for efficient revocation. Linkage authorities are a part of the pseudonym issuance. Then the RA shuffles the requests and sends all the requests to the PCA. Data sent to the PCA consist of requests which consists of encrypted (for PCA) pre-linkage values $(\text{plv}_1(i,j), \text{plv}_2(i,j))$, certificate validity time period $i$, index $j$ and cocoon public keys $B_\iota^A$ and $B_\iota^H$. Therefore each request from a vehicle to RA leads to many requests to PCA.

**Certificate generation:** The PCA receives a request. It is important that the RA should not be able to track a vehicle by linking pseudonyms to each other. In order to achieve that, for each request, the PCA generates a random integer $c$. Then the PCA obtains $C = cG$ and generates a certificate where the butterfly public key is $B_\iota^A + C$. In this way, the public key in the certificate is *hidden* from the RA. PCA generates linkage value which is XOR of the pre-linkage values and adds it to the certificate. PCA encrypts the certificate and the integer $c$ for the vehicle *with* $B_\iota^H$, and sends these back to RA.

**The RA relays certificates to vehicles:** The RA collects responses from the PCA and sends encrypted pseudonym certificates and corresponding private key construction values ($c$) to vehicles. The set of all the responses is called super batch.

**Information distribution:** After pseudonym issuance, each linkage authority has knowledge of initial linkage seed (generated normally only once) and pre-linkage value with corresponding $(i,j)$ values. PCA has knowledge of pre-linkage values from each linkage authority and their corresponding $(i,j)$ values, linkage value, cocoon public keys $B_\iota^A$ and $B_\iota^H$, certificate and hash of RA-to-PCA request. RA has knowledge of the enrolment certificate with its validity period, public key seeds ($A$ and $H$), expansion functions ($f_k$ and $f_e$), cocoon public keys $B_\iota^A$ and $B_\iota^H$ and the hash of RA-to-PCA request. After issuance none of the involved entities *alone* are able to link pseudonyms to enrollment certificates (i.e. resolve vehicles), or revoke pseudonyms.

### Pseudonym Resolution

Pseudonym resolution is not supported in SCMS. First, assume that a requester has an SCMS pseudonym. The requester can be for instance the misbehaviour authority. There is a linkage value attached to the pseudonym certificate. The requester sends the pseudonym to the PCA, which signed the pseudonym). The PCA retrieves from its database the RA-to-PCA hash with the pseudonym. The result is returned to the requester. Then, the requester sends the hash to the RA. The RA sends back the enrolment certificate. Then the requester has the corresponding long-term identity, the enrolment certificate. However, this last step where the RA sends the iden-

tity to the requester is not allowed in the SCMS, but it could be allowed with minor changes to the system if the requester is *trusted*.

SCMS meets all the security and privacy requirements listed in Sections 2.2.1 and 2.2.2, with the exception of accountability of vehicles, because pseudonym resolution is not supported by design. Perfect forward privacy cannot be evaluated, because, again, pseudonym resolution is not supported.

### 2.2.7 PUCA: A pseudonym scheme with strong privacy guarantees for vehicular ad-hoc networks

PUCA [21] is another scheme based on asymmetric cryptography and traditional PKI. It is designed with vehicles' privacy in mind. In PUCA, pseudonyms cannot be resolved. PUCA utilizes advanced cryptography for preserving users unconditional anonymity, also within the system. Because of that, PUCA is an interesting challenge from a pseudonym resolution point of view. Cryptographic methods are used besides for organisational separation to ensure users privacy. The authors write that they wanted to show that a scheme with unconditional vehicle anonymity can be designed. PUCA is built on top of the CAR 2 CAR Communication Consortium's basic pseudonym scheme [9] and it is compatible with the standardised approach to use PKI and pseudonymous certificates [2].

Participating entities in the system are the Long Term Certificate Authority (LTCA), the PCA, the RA and vehicles. With the LTCA, vehicles can register and obtain long-term credentials. From the PCA vehicles can obtain pseudonyms. Vehicles can send misbehaviour reports to the RA. The RA can revoke a vehicle. There is backend communication between LTCA and PCA. In a revocation process, the RA and PCA communicate with each other.

The technical details of PUCA are not discussed here because the details are not relevant in pseudonym resolution. PUCA can be modified to allow pseudonym resolution and needed modifications are discussed in Section 7.4.

## 2.3 Trusted Execution Environments

A TEE is a secure, integrity-protected environment, consisting of processing, memory, and storage capabilities [16]. A TEE is isolated from the Rich Execution Environment (REE), which is an environment containing a traditional computer operating system.

An important characteristic of TEEs is that software running inside the TEE is secure even if the REE is compromised (e.g. operating system compromise). Typical features of a TEE include *secure storage*, *isolated execution*, and *platform boot integrity* [6].

### Secure Storage

Secure storage covers the functionality of storing private data (e.g. private keys) in a TEE so that it will not be exposed to the REE but can be utilized through specific functions offered by an interface. The TEE supports a set of cryptographic algorithms, for instance for generating asymmetric key pairs and releasing only the public part to the REE. Secure storage covers also the functionality of *sealing*. Sealing denotes storing data outside the TEE so that the data is secret and integrity protected.

### Isolated Execution

Isolated execution means that it is possible to run a software in a TEE so that the related REE cannot affect to the execution control flow of the software. Note that it can be possible to interpret from the REE what is happening inside the TEE to some extent [10]. A TEE generally does not guarantee availability, because it can be interrupted by the REE. For instance, if a message is sent to a TEE from outside the TEE, delivery of a message could be intercepted by the corresponding REE or execution of the TEE could be delayed for so long time that the sender stops waiting for confirmation for the message. There is a strict security boundary between the TEE and the REE and interaction between the two can be done through the interface defined by the TEE developer.

### Intel SGX

A TEE can be implemented so that it is integrated to a CPU. An example of the approach is Intel SGX [28]. Note, in addition to the CPU, the other hardware and firmware on the platform must also support SGX. In SGX, *enclaves* are a central concept. SGX provides new CPU instructions for hardware enforced container generation. An enclave is a protected area in the application's address space.

With SGX it is possible to distribute a protected part of a software in the clear to a SGX capable machine and after setting up an enclave for the software attest remotely that the enclave's integrity is preserved. In the SGX programming model software is divided into two parts: trusted and untrusted parts. All instructions in the trusted part are executed in an enclave.

# 3 Problem Definition and Adversary Model

In this section the problem and an adversary model will be defined.

## 3.1 Problem Definition

For the problem definition, the following definitions are required:

**Definition 1** (Pseudonym holder-linkage)**.** A pseudonym $p$ is holder-linked to a vehicle $v$, if the $p$ is provisioned or issued for the $v$ by a PIA.

Note that in Definition 1, the PIA could be the vehicle itself, if the pseudonym scheme allows self-issuing of pseudonyms.

**Definition 2** (Pseudonym co-linkage)**.** Two pseudonyms are co-linked if they both are holder-linked to the same vehicle.

Accountability of vehicles appears to be in conflict with the pseudonym scheme requirements anonymity and unlinkability. From the conditional pseudonymity follows meeting of the pseudonym scheme requirement accountability of vehicles.

**Definition 3** (Pseudonym holder-linkage resolution)**.** Revealing the holder-linkage between a pseudonym and its holder.

**Definition 4** (Pseudonym co-linkage resolution)**.** Revealing the co-linkage between two or more pseudonyms.

The term *pseudonym resolution* (Definition 5) refers to both pseudonym holder-linkage resolution and co-linkage resolution.

Pseudonym holder-linkage resolution (Definition 3) leads to complete disappearance of anonymity with respect to the set of entities which are aware of the holder linkage. In other words, it leads to full pseudonym resolution. Pseudonym co-linkage resolution (Definition 4) leads to only partial loss of anonymity if it is assumed that none of the resolved pseudonyms were holder-linked already. In other words, generally, co-linkage resolution is a weaker form of pseudonym resolution.

**Definition 5** (Pseudonym resolution)**.** Pseudonym holder-linkage or co-linkage resolution.

Pseudonym resolution has a *time dimension* because each pseudonym is assumed to have a validity period. In pseudonym resolution there are at least two time frames. First is a pseudonym resolution time frame – pseudonym resolution within an identity system is possible in the time frame for an authorized entity. For example, a judge can authorize law enforcement authority to resolve a pseudonym within the next 7 days, after which time they will need to request a new authorization. Second is a target time frame

– only pseudonyms valid in the time frame can be resolved. Legislation may restrict time frames or it can set minimum or maximum lengths for the frames. For example, legislation may define a maximum pseudonym resolution time frame length or maximum age for target pseudonyms.

Pseudonym resolution must be accountable to be acceptable from political, social and legislative points of view. The legislative point of view depends on legislation areas, such as EU or USA. Accountability can be achieved by enforcing availability of pseudonym resolution metadata. Pseudonym resolution metadata can be for instance pseudonym resolution type (Section 4), the identity of an entity who gave permission, or order to resolve and justification for pseudonym resolution. Making statistical information about pseudonym resolution available to the public increases the transparency of the system. Further, the behavior of law enforcement authority and judges can be made transparent to some degree by making pseudonym resolution metadata statistics available. In other words, law enforcement authority and judges can be made accountable.

A pseudonym resolution target can be informed of pseudonym resolution [27]. Informing can be enforced to take place depending on the pseudonym resolution type or informing can be optional and require the authorization of a judge or other trusted authority.

**The problem** is to define an accountable pseudonym resolution system that

- considers the time dimension of pseudonym resolution
- makes pseudonym resolution accountable
- has capability to inform a target

### 3.2 Adversary Model

The adversary model considers adversaries having the standard Dolev-Yao capabilities [13], containing: intercept all communication messages, modify or drop messages, and send falsified messages. An adversary can collude with one or more vehicles or infrastructure entities in the system and control over multiple entities within the system.

The model is defined like this, because it takes into account internal and external adversaries and it has realistic message intercepting capabilities.

In the following, some adversarial entities and capabilities of the defined adversary are discussed.

The main aim of the adversary is to perform unauthorized pseudonym resolution, or non-accountable pseudonym resolution. In the following discussion, the term *pseudonym resolution service* denotes a service for resolving pseudonyms. Additionally it is assumed that the function which is allowed to request pseudonym resolutions is called LEA. The verb *alter* covers removing, editing, and adding data.

The following entities could be adversarial.

### Vehicles

Vehicles can interact with an identity management system and possibly, depending on implementation and integration, with the pseudonym resolution service. Therefore discussing threats following from malicious vehicles is relevant.

Assume that the adversary controls a vehicle which is a legitimate participant of an identity management system and that the adversary controls additionally a pseudonym resolution service component. If the identity management system supports misbehaviour reporting messages, the adversary could send a spoofed misbehaviour report of a target vehicle from the controlled vehicle to the identity management system so that it leads to usage of the controlled pseudonym resolution service component resulting in revealing data of the target to the adversary.

The adversary could try to send a false ID data from a controlled vehicle to the pseudonym resolution service via the PIA when it requests a new pseudonym set. The adversary could try to alter data in the pseudonym resolution service by commanding a controlled vehicle.

### Pseudonym Issuing Authority (PIA)

The PIA is a relevant malicious entity, since in many pseudonym schemes it is the most likely entity which could deliver linkage data of issued pseudonyms to the pseudonym resolution service.

A malicious PIA could try to send false linkage data to the pseudonym resolution service. When the PIA issues pseudonyms, it could send issued pseudonyms with some other vehicle ID to the pseudonym resolution service instead of the real vehicle ID of the real receiver of pseudonyms. Also, the PIA could just issue pseudonyms without sending any linkage data to the pseudonym resolution service. The controlled PIA could send all the data sent to the pseudonym resolution service also to the adversary. These issues are more likely issues of a related identity management system. But it does not mean that these threats should not be considered.

### Law Enforcement Authority (LEA)

Law enforcement authority is assumed to interact with the pseudonym resolution service.

A malicious law enforcement authority representative could try to do unauthorized pseudonym resolution by stealing credentials from an authorized law enforcement authority representative. That could lead to a situation, where some entity is accountable of unauthorized resolution. A controlled law enforcement authority representative could also try to get

unauthorized pseudonym resolution data from the pseudonym resolution service without stolen credentials or it may alter data by attacking against the pseudonym resolution service for instance by exploiting a vulnerability of it.

### Related Models

Raya and Hubaux [34] propose an adversary model for V2X. They divide different types of adversaries with the following classification:

- *Internal* vs. *External* An internal adversary is an authenticated member of a system, which is able to communicate with other members of the system.
- *Malicious* vs. *Rational* A malicious adversary wants to harm to a system and does not want personal benefits. A rational adversary is seeking personal profit and wants to achieve its goals as silently as possible (no more harm to a system than needed).
- *Active* vs. *Passive* An active adversary can generate packets or messages. Whereas a passive adversary can just eavesdrop on communication.
- *Global* vs. *Local* A global adversary controls many entities scattered widely across a system. A local adversary has control over a limited number of entities in a limited area.

The classification is also treated in [31]. With the classification, different types of adversaries can be denoted by abbreviations, for instance, I.R.A.G refers to an adversary which is Internal, Rational, Active and Global. The I.R.A.G adversary is the closest to the adversary model assumed in this thesis.

# 4 Requirements

## 4.1 Environment

Accountability of vehicles of a vehicle appears to be in conflict with anonymity and unlinkability. This apparent conflict is *settled* by making the judge accountable for mitigating anonymity or unlinkability. In other words, pseudonym resolution is made accountable. Another methods to add accountability to this process are informing a target of pseudonym resolution and make pseudonym resolution transparent to some degree by delivering statistics of pseudonym resolutions.

In the identity management system, pseudonyms contain validity time, and it is possible to send encrypted messages to vehicles by knowing their long-term identities. There can be one judge. A function which is allowed to request pseudonym resolution is called law enforcement authority. There can be one law enforcement authority representative.

## 4.2 Types of Pseudonym Resolution

Different types of pseudonym resolutions are listed in Table 1. The proposed solution in Section 5 is designed to be compatible with wide variety of resolution requests. Each of the requests in Table 1 are explained briefly.

| | |
|---|---|
| 1. | Given a pseudonym $p$, a law enforcement authority can request the corresponding holder-linked vehicle identity $v$ |
| 2. | Given a vehicle identity $v$ and a time frame $[t_1, t_2]$, a law enforcement authority can request the set of all co-linked pseudonyms $\{p_1, \ldots, p_n\}$ that are holder-linked to $v$ during that time frame |
| 3. | Given two pseudonyms, a law enforcement authority can request information (true / false) whether the given pseudonyms are co-linked |
| 4. | Given a pseudonym $p$ and a time frame $[t_1, t_2]$, a law enforcement authority can request the set of all co-linked pseudonyms $\{p_1, \ldots, p_n\}$ which are valid in that time frame |

Table 1: Resolution requests

**The first request** is for finding a holder vehicle for a pseudonym. The request demands resolving pseudonym holder-linkage.

**The second request** takes into account the time dimension of pseudonym resolution. The result of the second request is all co-linked pseudonyms which are valid in the time frame and which are holder-linked to the same vehicle.

**The third request** is used to determine whether two pseudonyms are co-linked or not. This could be extended to more than two pseudonyms, if required.

**The fourth request** is similar to the second request, but a pseudonym is given as an argument instead of a vehicle identity. In the fourth request, all the returned pseudonyms should be co-linked to the given pseudonym.

### 4.3 Requirements

**Requirement 1** (Authorised pseudonym resolution)**.** The law enforcement authority representative can request pseudonym resolution only if the representative is authorised by the judge. The list of possible types of resolutions is given in Table 1

**Requirement 2** (Target informing and pseudonym resolution justification data)**.** If pseudonym resolution happens, then the pseudonym *resolution justification data* from the accountable judge must be published. If the judge decides that the target should be informed, then, after time $t$ decided by the judge *only* holder $v$ for the $p$ *can* know that the $p$ was resolved and delayed pseudonym resolution justification data is published.

**Requirement 3** (Statistical pseudonym resolution information)**.** The system must provide public statistical information regarding all the pseudonym resolutions specifying the percentages that are co-linked vs. holder-linked and the percentage of pseudonym resolutions which have been informed.

With justification data, the judge can defend its decision to give permission to resolve. Or if the judge is malicious, the judge will be caught if justification data is not plausible enough or it is incorrect.

# 5 Solution Design

The solution is designed following the listed goals (below) and requirements presented in the previous section.

The solution design should:

- be compatible with existing V2X pseudonym schemes (Section 2.2)
- have a trusted or distributed resolution authority
- add or replace[2] accountability of vehicles to a scheme
- consider the time dimension of pseudonym resolution
- make pseudonym resolution accountable, in other words: make one or more entities accountable for pseudonym resolution
- make it possible that pseudonym resolution is transparent to some degree
- consider minimum disclosure
- maintain anonymity
- maintain unlinkability
- maintain perfect forward privacy

## 5.1 Participating Entities

Participating entities are vehicles, Pseudonym Issuing Authority (PIA), Accountable Pseudonym Resolution Service (APRS), Law Enforcement Authority (LEA), Judge (J), Public Forum (PF), and Checking ID Storage (CIDS).

In the system description, all but public forum and CIDS depicts multiple such entities. For instance, there are many vehicles involved in an identity management system. On the other hand, public forum and CIDS are designed to be global resources – possibly distributed. APRS is logically singular but can be replicated. Logically singular means that it does not matter to which replica of the APRS a request is sent, because the APRS is stateless.

Law enforcement authority depicts an entity which can request data from APRS – request pseudonym resolution, if it has permission to do so from a judge. Requested data can be for instance the real identity for a pseudonym, as defined in Table 1. APRS is the most critical element in

[2]Replacing accountability refers to a situation where an identity management system already meets the accountability of vehicles requirement, but possibly in a non transparent way.

the system because it stores all the sensitive information and decides what information to publish to the public forum and send to law enforcement authority. APRS is designed to be run in a Trusted Execution Environment (TEE). Therefore law enforcement authority can ask APRS to attest that its TEE is formed and behaving as expected. The public forum is for publishing public messages. There are two types of these messages: PFM1 and PFM2. A PFM2 is always related to a PFM1. Requirements for public forum can be found from Section 5.7. The CIDS is for publishing Checking IDs (CIDs) for vehicles.

## 5.2 Data Import to APRS

Since resolution is not time critical, linkage data should be available on demand or immediately from a database. In any case, linkage data should be available at some point. In this section, data import is divided into two cases for better compatibility to different pseudonym schemes.

### 5.2.1 With Identity-Pseudonym Linkage

Data import in case where an identity management system does know linkage between vehicle identities and pseudonyms. In this case, vehicle identity can be delivered by the identity management system. PIA sends message $[\{p_1, \ldots, p_n\}, E_{PK_{APRS}}(v)]^{\mathrm{S}}$ to the APRS, where S denotes a secure channel.

### 5.2.2 Without Identity-Pseudonym Linkage

Data import in case where an identity management system does not know linkage between vehicle identities and pseudonyms (e.g. PUCA). In this case, vehicle identity should be received from the vehicle itself. Data import functioning can be found from figure 3. Note that in the PUCA case the PIA is the PCA. The vehicle can ask the APRS to attest that its TEE is correct before sending the pseudonym request message.

## 5.3 Checking IDs for Vehicles

In this solution, CIDs are needed for efficient PFM2 checking for vehicles. A CIDs can be delivered to a vehicles by publishing and maintaining a CIDS. The CIDS could be in the form as follows. The storage can be thought as an array. A row for a vehicle in the array consists of a hash of the vehicle's long-term ID, a CID encrypted for the vehicle and optionally (see Section 5.5) a sequential numbers starting point encrypted for the vehicle (for example 634736). CIDs are signed by the APRS.

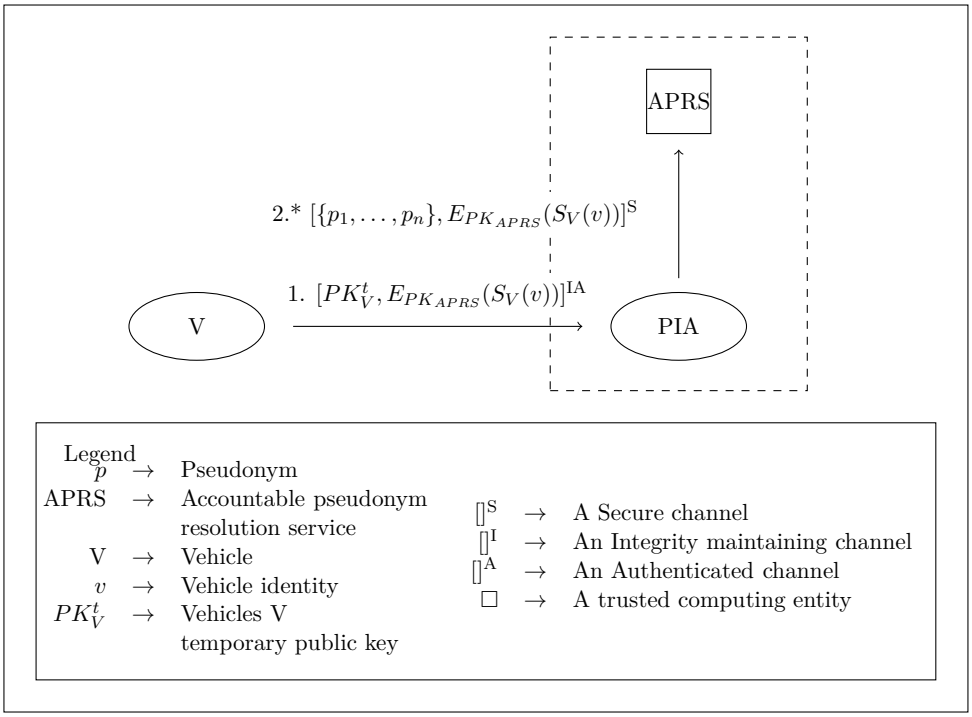


Figure 3: Data import

## 5.4 Generic Pseudonym Resolution System

### 5.4.1 System overview

An overview of the design of a generic pseudonym resolution system is shown in figure 4. A PKI can be used for authentication between instances of judge, law enforcement authority and APRS. Channels between law enforcement authority, judge, and APRS are considered to be secure. All messages will eventually be received (e.g. dropped messages will be resent until acknowledged).

### 5.4.2 System Interaction

The interaction between the entities is shown in Figure 4.

**Resolution request:** In the first message, a law enforcement authority requests permission to resolve from the judge. In the request, the law enforcement authority sends at the desired resolution types and two time frames in the variable $\alpha$. First time frame is a pseudonym resolution time frame – during this time frame the law enforcement authority is allowed to resolve (send pseudonym resolution requests to the APRS so that the APRS accepts a request). The second time frame is for limiting validity times of pseudonyms available in pseudonym resolution.

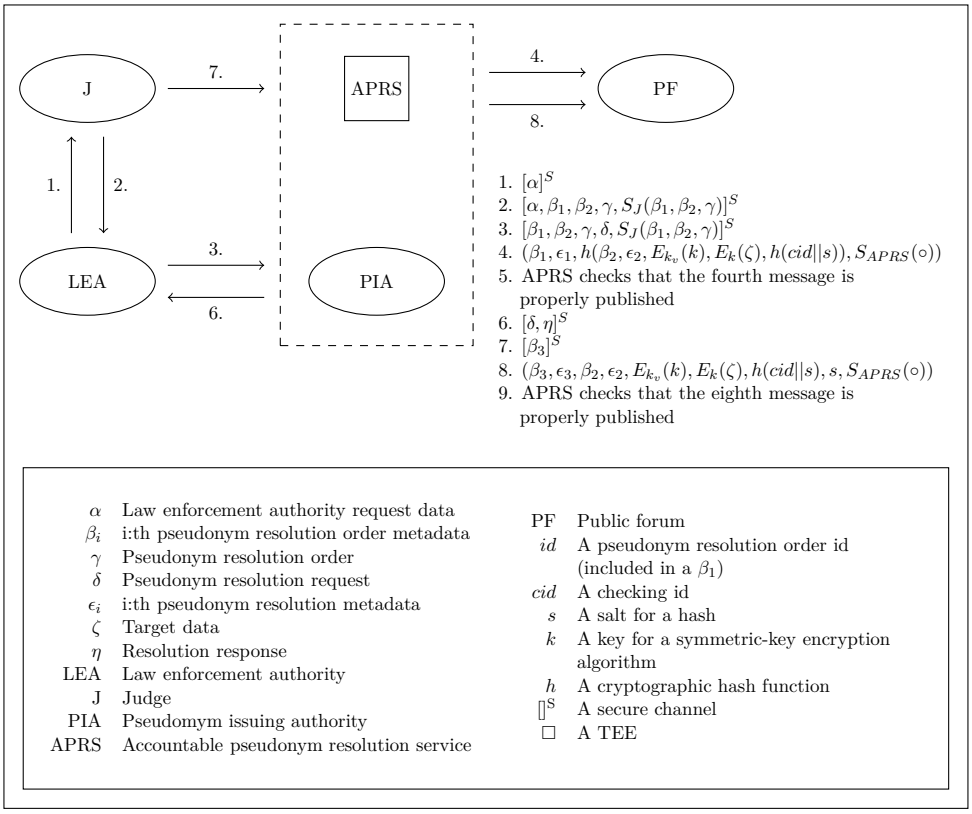


Figure 4: Generic pseudonym resolution system

**Resolution order:** The judge sends a response in the second message. The response consists of $\alpha$, $\beta_1$, $\beta_2$, $\gamma$ and signature of the last three. The signature is needed for law enforcement authority to be able to prove that it is allowed to resolve. $\beta_1$ contains pseudonym resolution metadata which should be revealed right after pseudonym resolution and should be visible for everyone. The metadata contains the pseudonym resolution order ID. $\beta_1$ can also contain for instance a vague reason for allowing pseudonym resolution. $\beta_2$ should be revealed only if the judge orders so. $\beta_2$ can contain pseudonym resolution metadata, which should be revealed only after investigations. $\beta_2$ contains at least the same ID (pseudonym resolution order ID) that is included in $\beta_1$, so that these two can be associated with each other. $\beta_1$ and $\beta_2$ are used for making the judge accountable. Variable $\gamma$ contains the actual pseudonym resolution order. The order consists of the same components as variable $\alpha$. Additionally, there should be an law enforcement authority identifier in the $\gamma$, because otherwise the order could be stolen and misused. Similar components in $\gamma$ and $\alpha$ can differ. For instance, the judge can decide to reduce the pseudonym resolution time. Thejudge can also decline a request completely.

**APRS request:** In the third message, the law enforcement authority sends a pseudonym resolution request to the APRS. The message contains $\delta$ – the request itself. $\delta$ contains a pseudonym resolution ID for making pseudonym resolution requests with the same order ID distinguishable. Law enforcement authority can use an order to resolve multiple times. Additionally, the third message contains $\beta_1$, $\beta_2$, $\gamma$ and judge's signature of the last three variables. The APRS checks that the law enforcement authority is allowed to make the request $\delta$ by comparing it to the order $\gamma$. Law enforcement authority's identity is compared to the identity in the order. APRS checks also that the pseudonym resolution ID is fresh.

**First public forum message:** The fourth message is published if pseudonym resolution happens. The message consists of order metadata $\beta_1$, delayed order metadata $\beta_2$, pseudonym resolution metadata $\epsilon_1$, delayed pseudonym resolution metadata $\epsilon_2$, a symmetric key $k$ encrypted for the target vehicle $v$, target data $\zeta$ encrypted with key $k$ and a hash of target vehicle's $v$ CID concatenated with a salt $s$. $\epsilon_1$ and $\epsilon_2$ contain the same public pseudonym resolution ID generated by the APRS. The APRS knows the linkage between pseudonym resolution IDs and public pseudonym resolution IDs. Though after publishing the fourth message the law enforcement authority knows the mapping also. Salt $s$ is for encrypting all the data except $\beta_1$ and $\epsilon_1$. $s$ is also important in protecting the CID $cid$. For efficiency reasons, the same $s$ can be used for all the PFM2 messages with the same order ID. This is not a security problem since, for each pseudonym resolution, a fresh public pseudonym resolution ID is generated, which is inside the $\epsilon_2$, which is inside the hash. The delay mechanism is implemented with the salt $s$. Optionally, the $s$ can be revealed in the eighth message. The whole message is signed by the APRS.

**APRS result:** If the fourth message is published properly, then the system can proceed to step 6. In the sixth message, the APRS delivers a response data $\eta$ to the law enforcement authority. Note that with the $\eta$, the original $\delta$ is delivered. The $\delta$ contains a pseudonym resolution ID and therefore the law enforcement authority can know which request is related to the response.

**Optional second public forum message:** Optionally, judge can send an order to reveal all the encrypted data in the fourth message. With the order, judge sends $\beta_3$, which contains the same order ID that is in $\beta_1$ and $\beta_2$. Additionally, it can contain something which judge would like to reveal to keep its accountability under observations of users of the system.

**Publishing second public forum message:** When the judge sends the seventh message, then the APRS publishes the eighth message. In the eighth message, salt $s$ is revealed, and therefore all which were encrypted with it. $\beta_3$ is revealed also. A vehicle can find out whether the message is for it by checking that the $h(cid||s)$ hash in the PFM2 is the same as a value of $h$ with its checking ID concatenated with the $s$ as the argument.

Additionally, the vehicle can find out the target data $\epsilon$ by decrypting it with the key $k$.

The message sequence chart of this interaction is shown in Figure 5.

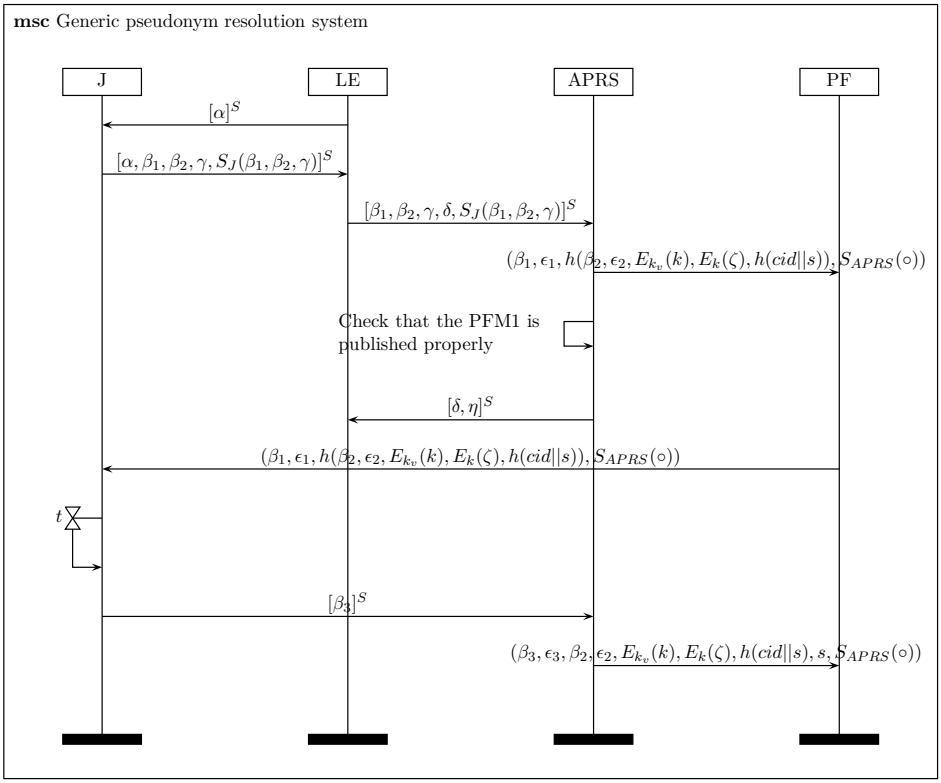


Figure 5: Generic pseudonym resolution system message sequence chart

### 5.4.3 Variable Specifications and Visibilities

Variable specifications for the variables $\alpha$, $\beta$,... can be found from Table 2. The variables are also for grouping data by visibility for participating entities or revealing time. Participating entities from a data visibility point of view are law enforcement authority, judge, APRS, a target vehicle, and all the other entities. Variable visibilities can be found from Table 3. In the visibilities-table "pre PFM1" denotes the time before the first public forum message and "post PFM1" correspondingly the time after the first PFM. Explanations for "pre PFM2" and "post PFM2" are similar. The value of (pre PFM1, $\beta_3$) is gray J. Gray color denotes the uncertainty of judge's knowledge of $\beta_3$ before the PFM1.

| Variable | Origin | Components |
|---|---|---|
| $\alpha$ | LEA | justification<br>pseudonym resolution types<br>pseudonym resolution time frame<br>pseudonym validity time frame |
| $\beta_1$ | J | order ID<br>justification<br>pseudonym resolution order metadata<br>• $h$(order ID\|\|LEA ID) |
| $\beta_2$ | J | order ID<br>delayed justification<br>delayed pseudonym resolution order metadata |
| $\beta_3$ | J | order ID<br>delayed additional justification<br>delayed additional pseudonym resolution order metadata |
| $\gamma$ | J | LEA ID<br>pseudonym resolution types<br>pseudonym resolution time frame<br>pseudonym validity time frame<br>Js signature over the message |
| $\delta$ | LEA | pseudonym resolution ID<br>pseudonym resolution request arguments |
| $\epsilon_1$ | APRS | public pseudonym resolution ID<br>pseudonym resolution metadata<br>• judge ID |
| $\epsilon_2$ | APRS | public pseudonym resolution ID<br>delayed pseudonym resolution metadata |
| $\epsilon_3$ | APRS | public pseudonym resolution ID<br>delayed additional pseudonym resolution metadata |
| $\zeta$ | APRS | Target informing data |
| $\eta$ | APRS | pseudonym resolution result |

Table 2: Message variable specifications. Lines with bullets are obligatory sub-items for items above bulleted items.

### 5.5 More Efficient PFM2 Checking

In Section 5.4.2 the PFM2 message checking functionality for vehicles is rather inefficient because of the aggregate amount of used resources. In this section a solution for the resource usage problem is presented.

Bloom filters can be utilised to achieve even more efficient PFM2 checking. Optionally, a Bloom filter with capacity $n$ and probability $p$ can be published once in $n$ PFM2 messages containing all $n$ $h(cid||n_{seq})$ hashes for published PFM2 messages. Within a published Bloom filter, also a public

| Visibility | pre PFM 1 | post PFM 1 & pre PFM 2 | post PFM 2 |
|---|---|---|---|
| $\alpha$ | LEA, J | | |
| $\beta_1$ | J, LEA, APRS | *all* | |
| $\beta_2$ | J, LEA, APRS | | *all* |
| $\beta_3$ | J | J, APRS | *all* |
| $\gamma$ | J, LEA, APRS | | |
| $\delta$ | LEA, APRS | | |
| $\epsilon_1$ | APRS | *all* | |
| $\epsilon_2$ | APRS | | *all* |
| $\epsilon_3$ | APRS | APRS | *all* |
| $\zeta$ | APRS | | *target* |
| $\eta$ | APRS | LEA, J | |

Table 3: Message variable visibilities

pseudonym resolution ID interval should be published. If the more efficient PFM2 checking is used, then the generated public pseudonym resolution IDs should form a rising ordered list. Sequential numbers $n_{seq}$ can be for example in the set of two-byte integers $\{0, \ldots, 65535\}$. The cardinality of the set determines the interval for publishing a new CID for a vehicle. *No value in the set should be re-used with the same CID.* Otherwise, linking values $h(cid||n_{seq})$ with the same *cid* is possible – an adversary can find out that a vehicle with a *cid* is resolved at least cardinality of the set times. Therefore a new fresh CID should be delivered for a vehicle when all of the sequential numbers are used. Using the set $\mathbb{N}$ is not practical from the technical point of view.

If a vehicle needs to check that is there a PFM2 for it, it should know its current CID and the corresponding sequential number. The vehicle starts the checking process by downloading the oldest unchecked Bloom filter. Then the vehicle checks whether the filter contains the $h(cid||n_{seq})$ or not. If it contains it, then the vehicle checks all the PFM1 messages in the interval. If there is no matching PFM1 for the vehicle, then the match was a false positive. If there is a match in the interval, then the vehicle should update its sequential number to the next by increasing the old one by one (modulo the cardinality of the set) or get a new CID with a corresponding initial sequential number if there are no sequential numbers left for the CID. Then the vehicle can continue by downloading a new oldest unchecked filter and the process starts over.

Counting Bloom filters [19] can be used also. In that case, a vehicle can know before going through a PFM1 message interval that how many PFM1 messages there are for it. Knowledge of the number of matching PFM1 messages leads to more efficient checking because all the PFM1 messages in

an interval are not needed to go through in average. Counting Bloom filters are in general larger in size (bytes) than traditional Bloom filters without counting features.

## 5.6 Variables For Different Pseudonym Resolution Requests

In this section, it is presented how the proposed solution can be used for resolutions listed in Table 1. Resolution types are taken fromOB Table 1.

In Table 4, the column $\epsilon_1$ is critical for avoiding leakage of unauthorized linkage data. Even publishing a number of some items can reveal information from which one can infer unauthorized results.

For the third pseudonym resolution request in Table 1 there should be distinct PFMs with distinct public resolution IDs in the case where the pseudonyms are not co-linked. Otherwise, the targets could find afterwards that they were resolved in the same resolution.

| Type | $\delta$ (pseudonym resolution request) | $\epsilon_1$ (pseudonym resolution metadata) | $\zeta$ (target data) | $\eta$ (resolution result) |
|---|---|---|---|---|
| 1 | (type: 1, resID, $p$) | (type: 1, pubResID, jID) | $(p)$ | $(v)$ |
| 2 | (type: 2, resID, $v$, $[t_1, t_2]$) | (type: 2, pubResID, jID, $n$) | $((p_1, \ldots, p_n), [t_1, t_2])$ | $(p_1, \ldots, p_n)$ |
| 3 | (type: 3, resID, $(p_1, p_2)$) | for each holder $h$ $\epsilon_{1_h}$ = (type: 3, pubResID: getFreshID(), jID) | for each holder $h$ $\zeta_h = (p_h)$ | $\eta = (\text{holder}(p_1) = \text{holder}(p_2))$ |
| 4 | (type: 4, resID, $p$, $[t_1, t_2]$) | (type: 4, pubResID, jID, $n$) | $((p_1, \ldots, p_n), [t_1, t_2])$ | $(p_1, \ldots, p_n)$ |

Table 4: Variables for different pseudonym resolution requests

## 5.7 Requirements for the PF and the CIDS

Both the public forum and CIDS have the same requirements which are discussed in this section. Integrity protection of the public forum is important, because it is part of the mechanisms of making the judge accountable, publishing metadata of resolutions, and publishing target data.

**Integrity protected:** Appended values cannot be altered afterwards.

For efficiency and plausibility reasons the public forum should be appendable only by the APRS. If other entities were allowed to append, then searching for a PFM2 for a vehicle could be inefficient. Additionally, an entity could add the same PFMs to the public forum multiple times leading to

false statistics and the accountable judge being accountable of non existent resolutions.

**Append-only by defined entities:** Only defined entities can append values.

In the fifth step of the generic pseudonym resolution system (Figure 4) it should be possible to check that a PFM is published.

**Proof of publishing:** Given a proportion $p$, it must be possible to check that a value is appended so that the proportion $p$ of the legitimate participants of a related identity management system *can* read the appended value.

## 5.8 Compatible Pseudonym Schemes

A pseudonym scheme is compatible with the proposed solution if:

- for each pseudonym, the corresponding vehicle identity is available before a query or on demand,
- for each vehicle identity and a time frame, the corresponding pseudonyms are available before a query or on demand,
- for each vehicle it is possible to encrypt a given message for it by knowing its identity, and
- each vehicle can sign a given message and the signature can be verified by knowing the signers identity.

The SCMS is compatible with the solution with small changes. More details are given in Section 7.4.

# 6 Implementation Guidelines

In this section implementation guidelines for the APRS and implementation options for the public forum are discussed.

## 6.1 Implementation Guidelines for the APRS

Since in this solution the APRS has all the linkage data, avoiding side channels in the APRS is among minimum requirements for an implementation.

### Avoiding Side Channels

Using the Intel SGX TEE it is possible for example to enforce that the PFM1 is published before a pseudonym resolution result is sent to the requesting LEA. There are side channel attack against Intel SGX. The authors of [10] present a cache-based side channel attack against Intel SGX which uses the so called Prime+Probe cache monitoring technique. The authors present a full 2048-bit RSA key recovery from an SGX enclave with the attack. Another side channel approach is to infer control flow of a program by causing page faults [44].

An implementation must also take into account possible side channels caused by database usage. For instance, if a law enforcement authority sends a pseudonym co-linking request which is approved by a judge, then later, after the request is handled, no entity, including the law enforcement authority, should be able to get pseudonym holder-linkage (Definition 1) information for the pseudonym set without a permission from a judge, using some information from the pseudonym co-linking handling process. Logically, a pseudonym resolution system should internally link a pseudonym to its holder in order to be able to tell that two pseudonyms are co-linked to each other. That holder-linkage information should not leak. Not even a row number containing holder information should leak – it is possible that the row number can be used in getting some linkage information in an unauthorized way. The problem with leaking database row numbers is that if an adversary knows a database row number for a holder, it can later send just co-linking requests and get holder-linkage data at the same time. Leaking database row numbers violates perfect forward privacy.

Since pseudonym resolution is not time critical, one possible solution is to use a *carousel* [42]. In the carousel approach, database rows are circulated in the TEE so that there are for example 1000 rows available at the same time in the TEE. The carousel approach helps the situation but does not fix it completely [10], [44].

Another approach is to use an *oblivious RAM* scheme [23].

## 6.2 Implementation Options for the Public Forum/CIDS

The both public forum and CIDS can be implemented on the practical level using a same database.

**Issues of publishing public forum messages:** The issue of checking whether a PFM is published is not trivial to solve since the adversary could spoof that a message is published. In other words, the adversary could deliver false proof of publishing. Logically, the adversary can surround the APRS so that it intercepts all the outgoing messages and spoofs all the ingoing messages to deceive the APRS. A simple, yet effective solution is that the public forum is a trusted party which sends a confirmation back to the sender when it receives a message. In that case, the sender can trust that at least the message was received by the public forum. Additionally, authorizing vehicles to confirm publishing messages helps in avoiding false proofs of publishment.

**A solution:** In the following, a public forum and CIDS implementation is presented which meets the requirements integrity protected, append-only by defined entities, and proof of publishing, utilizes confirmation messages of vehicles and the public forum, and where the public forum is trusted. The trustworthiness of the public forum can be achieved by using a TEE. When the public forum receives a message from the APRS, it stores the message, publishes the message, generates a signature over a hash of the message and stores the signature. Some vehicles read the published messages. The vehicles sign the hashes of the published messages with their long-term keys and send the signatures to the public forum. The public forum stores and publishes the received signatures within the corresponding message from the APRS. The APRS reads all the signatures for a message from the public forum. If there is a valid signature from the public forum and *enough* signatures from distinct vehicles, the APRS can conclude that the message is published properly and continue to the sixth step in Figure 4. The APRS sends a message to the public forum to inform that the PFM record can be marked as confirmed. By doing that, vehicles will not do useless work.

**The motivation of vehicles:** The question is why a proportion of vehicles could be *motivated* to spent their resources by confirming PFMs. For example, victims of transportation accidents where a party escapes from the scene trying to hide its identity could be motivated to spent a small amount of resources for keeping the system functional. In addition, it is safe to assume that the majority of vehicles do not want to be target of uncontrolled surveillance. Confirming PFMs supports accountability of the system mitigating uncontrolled surveillance. Therefore a proportion of vehicles could be willing to confirm PFMs.

**Meeting requirements:** The requirement proof of publishing is met with the introduced steps and the requirement append-only by defined entities can met by restricting the public forum to accept only messages signed

by the APRS. Therefore only the requirement integrity protected remains. The integrity protection is possible to achieve by putting the hash of the preceding PFM to each PFM by the APRS. By doing that, a vehicle could notice if a published PFM is removed or altered afterwards. Anyone can download all the PFM and check that nothing is altered afterwards.

Another public forum implementation option is for example a blockchain.

# 7 Evaluation

In this evaluation section, an extended adversary model, security evaluation, and deployability of the service are discussed. In the security evaluation, meeting solution requirements and resistance against the adversary model (Sections 3.2 and 7.1) are discussed.

Requirement 1 in Section 4.3 is met by the proposed solution withuout its authorization part as presented in Table 4. Requirement 3 in Section 4.3 is met by the proposed solution because in the solution a PFM1 contains variable $\epsilon_1$ (Figure 4), which contains pseudonym resolution metadata (Table 2), which can contain resolution type as presented in Table 4. Additionally, the knowledge whether a target is informed about a resolution can be inferred from the existence of corresponding PFM2 for the PFM1. The PFM1 is published in any case when pseudonym resolution happens and the corresponding PFM2 is published if the judge decides to inform the target.

## 7.1 Extended Adversary Model

In this section, the adversary model (Section 3.2) is extended to cover the proposed solution. For the sake of simplicity, public forum and CIDS are called just public forum.

### Vehicles

A vehicle could try to read information for some other vehicle from the public forum. A vehicle could try to alter data in the public forum.

### Law Enforcement Authority (LEA)

A law enforcement authority representative could try to steal an order for some other law enforcement authority representative and use it to request unauthorized pseudonym resolutions. A law enforcement authority representative could try to infer some extra information that the law enforcement authority representative is not authorized to get by reading public data published by the APRS. A law enforcement authority representative could try to use stolen credentials of some other law enforcement authority representative to ask for permission to resolve from a judge with spoofed justification data and then use the same credentials to send resolution requests to the APRS.

### Judge

A judge could try to authorize a law enforcement authority representative to resolve with a fake identity or a stolen identity. In other words, authorizing some entity without being accountable. A judge could try to convince that it is not accountable of some resolution permission or resolutions done with

the permission. A judge could try to alter published PFMs for getting rid of its accountability.

#### Public Forum

The public forum could try to spoof proof of publishing. The public forum could try to alter added values.

#### Accountable Pseudonym Resolution Service (APRS)

The adversary could try to accept unauthorized resolutions by controlling the APRS. The adversary could try to avoid publishing PFMs when resolving by controlling the APRS. The adversary could try to intercept linkage data import messages by controlling the APRS so that some pseudonyms of a vehicle are not resolvable.

### 7.2 Security Evaluation

First, Tamarin prover model and a Tamarin prover security proof are discussed. The model and the proof are then used in justifying meeting the requirements and resistance against the adversary model.

#### 7.2.1 Tamarin Prover Model

"*The Tamarin Prover is a security protocol verification tool that supports both falsification and unbounded verification in the symbolic model. Security protocols are specified as multiset rewriting systems and analysed with respect to (temporal) first-order properties and a message theory that models Diffie-Hellman exponentiation combined with a user-defined subterm-convergent rewriting theory.*" [7].

**Overview of Tamarin:** The central parts of the Tamarin system are *multiset rewriting rules* and *multiset of facts*. The multiset denotes a set which can hold multiple instances of the same element. All *facts* are of the form `F(t1,...,tn)` where the `F` part is the symbol of a fact and `t1,...,tn` are terms of a fact. All the facts with the same symbol should have the same arity. The rewriting rules are of the form `[] -[]-> []` where each of the square brackets can contain many facts or no facts at all. For instance `[F1(a,b)] -[F2(b)]-> [F3(a)]` is syntactically a correct rule declaration in Tamarin. A rule can execute if the multiset of facts contains all the facts in the leftmost square brackets. If there is the exclamation mark before a fact, like `!F(t)`, it means that the fact is *persistent*. It means that the fact can be *consumed* by a rule without taking the fact away from the multiset. In turn, if a fact is not persistent, then it is taken away from the multiset when a rule consumes it. For instance, if the rule `[F1(a,b), !F2(a)] -[F3(b)]-> [F4(a)]` executes, one fact element `F1(a,b)` is taken away from the multiset but none of type `!F2(a)`. If a fact symbol is marked as permanent, it should

be marked permanent everywhere. In the same model, having facts `!F(a)` and `F(b)` is forbidden. Note that different term symbols can be used with the same fact symbol. If a rule executes, all the facts in the rightmost square brackets are added in the multiset. All the rewriting rules form a *multiset rewriting system*. Further, the multiset rewriting system defines a *transition system*. A state in Tamarin is a set of facts in the multiset. A transition is an execution of a rule. A transition can be *labeled* by adding one or more facts to the midmost brackets of the corresponding rule. Facts in the midmost brackets are called *action facts*. A *trace* is a chain of action facts. In other words, a chain of transitions.

**The model:** The Tamarin Prover model is in Appendix A. The model is explained in the following. *From now on, all the section references with the* `typewriter font` *are references to the model sections inside Appendix A.* The model is available also on [39].

**Communication channels in the model:** In the model, a secure channel is assumed in many protocol steps. A secure channel is modelled in Section `1.2.1 Secure channel`. The channel is authentic and confidential. Both sides are authenticated. The adversary can delay message delivery, but eventually messages are delivered. The adversary can also replay messages an unlimited number of times. The delay mechanism is modeled by putting a *permanent fact* `!S($A,$B,x)` to the *multiset of facts* by the rule `Chan_Out_S` on line 130. In the rule `Chan_In_S` on line 135, the fact `!S($A,$B,x)` can be read immediately, but delays are also possible. The replay is modeled at the same time. Permanent facts are not removed from the multiset and therefore the rule `Chan_In_S` can *consume* the fact `!S($A,$B,x)` multiple times.

**Number of participants:** The generic resolution system and the underlying identity management system are modeled so that there can be arbitrarily many vehicles. For the sake of simplicity, there are only one instances of the APRS and judge. There two law enforcement authorities. One of them is malicious and the other can leak information to the malicious law enforcement authority.

**CID delivery:** The CID delivery is modeled in Section `1.3 CID delivery`. In the rule `APRS_1` on line 144 the APRS sends the message CIDD1 to the CIDS. That is denoted in the rule by the action fact `APRS_sen_CIDD1_to_CIDS(~cid,pkV_lt)`. The same convention is used in all the rules and the convention follows the next template: `participant1_'sen'|'rec'_messageID_'to'|'from'_participant2()`. In the template, '|' denotes logical OR. More explanations on the convention are in Section `Abbreviations and action fact naming conventions`. With the sent message, a CID for a vehicle is delivered. In the rule `CIDS_1` on line 155 the CIDS receives the CID sent by the APRS. In the rule `V_1` on line 164 the vehicle finds the CID for it and stores it for checking PFMs.

**Data import:** The data import is modeled in Section `1.4 Data import`.

In the data import, the PIA sends a fresh pseudonym to the APRS. The secure channel is used.

**Overview of the generic resolution model:** The generic resolution system (Figure 4) is modeled in Section `1.5 Generic resolution system`. Note, that the rules in the section are labeled with action facts where the `messageID` is of the form `GRS`$i$. The abbreviation GRS stands for generic resolution system and the variable $i$ denotes steps in Figure 4. The model allows many resolution permission requests from the law enforcement authority. It allows only one response to a resolution permission request (the adversary can replay messages, but the law enforcement authority receives the same message only once). The restriction is modeled by putting the `UniqueFact(<'LEA_2',a,ordID>)` action fact to the rule `LEA_2` on line 224. Tamarin restricts itself to consider only the traces where the action fact occurs only once. The name `UniqueFact` therefore means "unique in a trace". The restriction is defined in Section `Restrictions`.

**From an order request to resolution:** The law enforcement authority sends a resolution request in the rule `LEA_3` on line 233. In the model, the law enforcement authority sends an pseudonym as a resolution argument for demonstrating and proving functionality of the model. Including a pseudonym in the variable $\delta$ is modeled by putting the pseudonym inside less-than and greater-than signs with the variable $\delta$ (`<~d,~resID,pseu>`). In the model, the law enforcement authority can send multiple resolution requests with the same resolution order. The model restricts the APRS from receiving a resolution request multiple times. This restriction is set in the rule `APRS_3` on line 247 as the action fact `UniqueFact(<'APRS_3_ordID_resID',ordID,resID>)`. From among the resolution arguments and the database query results the APRS chooses the target long-term identity. In the rule `APRS_3`, the target long-term identity comes from the fact `!APRS_2(pseu,pkV_lt)`. The APRS must verify validity of the resolution request. It verifies that the order is signed by a legitimate judge and that the sender identity is in the order. Comparing the request to the order is not modeled. If all the checks pass, the APRS sends a PFM1 to the public forum.

**From checking publishment of the PFM1 to sending an resolution result:** The public forum receives the PFM1 in the rule `PF_1` on line 273. The public forum check, that the message is from the APRS by verifying the signature in the message. A PFM should be stored in the public forum only once. That restriction is modeled with the action fact `UniqueFact(<'PF_1',pub_resID>)` in the rule `PF_1`. The APRS checks that the PFM1 is published properly. That is modeled by demanding the fact `!PF_storage(promise)` in the rule `APRS_4`, which is in the multiset only if the public forum received the PFM1. The APRS continues by sending the resolution result to the law enforcement authority and a notification to the judge. In reality the notification is not needed, but the judge knows

that how long the order can be used to resolve and can decide after the order validity period whether to order to send PFM2 messages or not. The law enforcement authority receives a response from the APRS for the resolution request in the rule `LEA_4` on line 299. The law enforcement authority stores the resolution results. The law enforcement authority also verifies that the answer really comes from the APRS. This is modeled in the rule `LEA_4` with the action fact `Eq(verify(sig,«d,resID>,eta>,pkAPRS),true)`. The `Eq` restriction is presented in Section `Restrictions`

**Judge decides to inform the target:** The judge sends an order to send the PFM2 in the rule `J_3` on line 318. The APRS receives the order in the rule `APRS_5` on line 325. Note that when the judge sends an order to send the PFM2, all the PFM1 messages in the public forum should receive a corresponding PFM2. It means that the APRS should repeat the step 8 in Figure 4 possibly many times. The repeating is modeled by allowing receiving the message from the judge in the rule `APRS_5` by *not* restricting the rule from executing in a trace multiple times with an `ordID`. Note that executing the rule multiple times in a trace with the same `resID` *is* restricted.

**Vehicles searching target data:** Vehicles can find target data for them in the rule `V_2` on line 353. Action fact `Eq(hash_cid,h2(<cid,s>))` in the rule follows the checking procedure presented in Section 5.4.2: a vehicle checks whether the hash $h(cid||s)$ (`hash_cid`) matches with the hash of the CID of the vehicle and the $s$ (`h2(<cid,s>)`). Additionally the vehicle can check that that the PFM2 contains really data which was generated for the PFM1 by checking that the hash in the PFM1 equals to the hash of $\beta_2, \epsilon_2, E_{k_v}(k), E_k(\zeta), h(cid||s)$ from the PFM2
(`Eq(hash,h1(<b2,e2,enc_k,enc_z,hash_cid>))`).

**Leaking orders:** In Section `1.7 Malicious entities`, the model takes into account the possibility of leaking resolution orders to law enforcement authorities which are not authorized to resolve with the orders. There are three spots where an order can leak: the judge, the law enforcement authority, and the APRS. Note that channels between the three are assumed secure. The law enforcement authority leaks an order in the rule `LEA_5` on line 382. Leaking the order and using it is modeled by leaking the order to the adversary in the rule `LEA_5`
(`Out(«b1,ordID>,b2,<g,pkLE>,sign(«b1,ordID>,b2,<g,pkLE»,skJ)>)`), leaking the private key of the malicious law enforcement authority to the adversary in the rule `Malicious_LEA_1` on line 392, and letting the adversary to decide what data the malicious law enforcement authority sends to the APRS in the rule `Malicious_LEA_2` on line 398.

### Tamarin Prover Security Proof

The security of the solution can be proved with lemmas in Section `2. Lemmas`. The actual proof can be generated by installing the Tamarin Prover [7],

downloading the model on [39], and running command `tamarin-prover accountable_pseudonym_resolution_service.spthy --prove`. The proof is also available on [40]. Because the commandline option `--heuristic` is not used in generating the proof, the default heuristic, 'smart', is used. Details of the heuristic can be found from the Tamarin manual on [7] or from [37].

The proof is relevant against the adversary model since Tamarin contains a Dolev-Yao adversary. The adversary controls the network and can delete, inject, modify and intercept messages on the network. In the model, all facts of the form `Out(message)` in the multiset are *known* by the adversary. For instance, if the rule `[F1(a,b), !F2(a)] -[F3(b)]-> [Out(a)]` executes, the adversary will know all readable contents of the variable `a`.

In Section `2.1 Sources lemmas` the lemmas are for helping the *Tamarin to* know *where a variable is originated. In other words: where a variable is fresh.* An interested reader can refer to the Tamarin manual available on [7].

With lemmas in Section `2.2 CID delivery`, the CID delivery can be proved to be functional and secure. Functionality lemmas are in Section `2.2.1 Functionality` and security lemmas are in Section `2.2.2 Security`. The same sectioning style is used in Section `2.3 Data import` and Section `2.4 Generic resolution system`.

The main motivations for this section is to show that the proposed design in Section 5 (Figure 4) maintains variable visibilities presented in Table 3 and that the security lemmas are relevant. Then these variable visibilities, the variable specifications (Table 2), and the security lemmas can be used in the following sections.

The most important lemma in Section `2.4 Generic resolution system` is the `the_system_is_functional` lemma on line 559. The reason for that is that if the lemma does not pass, then many security lemmas could pass trivially. For example, if a lemma can be written in natural language *"for all traces where a law enforcement authority representative receives a resolution result, it is true that the corresponding PFM1 is published before the law enforcement authority receives the result"* and there are no traces in the model where *"a law enforcement authority representative receives a resolution result"*, then the lemma is trivially true because there are no traces which can contradict the latter part *"it is true that the corresponding PFM1 is published before the law enforcement authority receives the result"* of the lemma. Because the lemma `the_system_is_functional` passes, it means that the model is functioning and that the security lemmas are relevant.

Table 5 shows which lemmas can be used in proving that the model maintains message variable visibilities presented in Table 3.

Note that lemmas in Table 5 consider message variable visibilities in less granular way than Table 3. However, the model and lemmas consider the most important visibility boundary – the adversary visibility boundary.

| Variable | Lemmas |
|---|---|
| $\alpha$ | `a_is_not_known_by_the_adversary` on line 827 |
| $\beta_1$ | `b1_is_not_known_by_the_adversary_before_a_PFM_1` on line 847 |
| $\beta_2$ | `b2_is_not_known_by_the_adversary_before_PFM_2` on line 876 |
| $\beta_3$ | `b3_is_not_known_by_the_adversary_before_PFM_2` on line 900 |
| $\gamma$ | `g_is_not_known_by_the_adversary` on line 922 |
| $\delta$ | `d_is_not_known_by_the_adversary` on line 942 |
| $\epsilon_1$ | Not relevant since $\epsilon_1$ is generated in the APRS just before sending it to the public forum. |
| $\epsilon_2$ | `e2_is_not_known_by_the_adversary_before_PFM_2` on line 969 |
| $\epsilon_3$ | Not relevant since $\epsilon_3$ is generated in the APRS just before sending it to the public forum. |
| $\zeta$ | `z_is_not_known_by_the_adversary` on line 995 |
| $\eta$ | `eta_is_not_known_by_the_adversary` on line 1009 |

Table 5: Message variable visibility lemmas

**Evaluating Requirement 1 (Authorised Pseudonym Resolution)**

With lemma `only_authorized_LEA_can_resolve` on line 1067 it can be proved that the model and therefore the design allows only authorized pseudonym resolution.

The lemma `only_authorized_LEA_can_resolve` can be written in natural language as *"whenever a LEA receives a resolution result and no resolution requests have been leaked, the judge authorized the resolution and the LEA used the authorization to resolve"*. The actual proof for the lemma can be found from [40]. In Tamarin, proving the lemma `only_authorized_LEA_can_resolve` takes 19 steps.

The lemma demands that whenever the action fact `LEA_rec_GRS6_from_APRS(<d,resID,pseu>,<eta,pkV_lt>)` occurs in a trace, in the same trace should occur also action facts `J_sen_GRS2_to_LEA(a,<b1,ordID>,b2,<g,pkLEA>,sig)` and `LEA_sen_GRS3_to_APRS(<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sig,sig_3)`. Because there are the same variable `ordID` in the two action facts, these action facts are bound together. Further, there are the same variable `resID` in action facts `LEA_rec_GRS6_from_APRS(<d,resID,pseu>,<eta,pkV_lt>)` and `LEA_sen_GRS3_to_APRS(<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sig,sig_3)`, so these facts are also bound together. Therefore facts `LEA_rec_GRS6_from_APRS(<d,resID,pseu>,<eta,pkV_lt>)` and `J_sen_GRS2_to_LEA(a,<b1,ordID>,b2,<g,pkLEA>,sig)` are bound together. That means that in the model, for all traces, if a resolution happens the judge authorized it by sending an order.

**Evaluating Requirement 2 (Target Informing and Pseudonym Resolution Justification Data)**

Meeting Requirement 2 in Section 4.3 is discussed in this section.

In the proposed solution, pseudonym resolution can happen only with a valid order from a judge. With the order, the judge delivers justification data inside variable $\beta_1$ (Figure 4 and Table 2). The justification data cannot be altered, because the judge signs variables $\beta_1$, $\beta_2$, and $\gamma$. The APRS checks the signature before it approves a resolution request or the justification data. The APRS is enforced to publish the justification data before it sends a resolution result to a requester law enforcement authority representative. Therefore the pseudonym resolution justification data is published if pseudonym resolution happens and the data is from an accountable judge.

With lemma `res_leads_to_publishing_a_PFM1` on line 759 it can be proved that the model enforces publishing a corresponding PFM1 when resolution happens.

The lemma `res_leads_to_publishing_a_PFM1` can be written in natural language as *"whenever the LEA receives a resolution result, the corresponding PFM1 was published"*. Proving the lemma takes 13 steps.

The lemma demands that whenever the action fact `PF_rec_GRS4_from_APRS(<b1,ordID>,<e1,pub_resID>)` occurs in a trace, in the same trace should occur also action fact `PF_rec_GRS4_from_APRS(<b1,ordID>,<e1,pub_resID>)` and that the latter fact occurs *before* the former fact. These two facts are bound together via variables `resID`, `pub_resID`, and `ordID`. That means that in the model, for all traces, if resolution happens, then a corresponding PFM1 was published.

A judge can send an informing order to the APRS. The order consists of variable $\beta_3$, which contains for example order ID (Table 2). The APRS is enforced to publish the corresponding (to the order ID) PFM2. The PFM2 contains delayed justification data from the judge inside variable $\beta_2$. Therefore delayed pseudonym resolution justification data is published if the judge decides so.

Note that it cannot proved *for all traces* that when the judge sends an order to publish it leads to publishing the corresponding PFM2. The reason is that in a trace, the adversary can intercept all the messages from the judge. Lemma `Js_order_to_send_PFM2_messages_leads_to_publishing_a_correct_PFM2` on line 628 shows that it is *possible* in the model that an order leads to publishing a PFM2.

The remaining part is to show that only the target can know that its pseudonym has been under resolution. In the proposed solution, target data is published inside variable $\zeta$ in a way that only the target can read it. The target data is encrypted with a symmetric key. The symmetric key is encrypted with the long-term key of the target vehicle. Security of target data

variable $\zeta$ can be proved with lemma `z_is_not_known_by_the_adversary` on line 995 in the model.

Lemma `z_is_not_known_by_the_adversary` can be written in natural language as *"whenever the APRS publishes a PFM2 it should not be that the adversary knows the target data zeta.* The lemma makes sure that the adversary cannot know target data.

Lemma `a_vehicle_can_read_target_data_published_for_it` on line 740 can be used to prove that a target *can* find target data for it.

### Evaluating Requirement 3 (Statistical Pseudonym Resolution Information)

With lemma `res_leads_to_publishing_a_PFM1` on line 759 it can be proved that whenever a resolution happens, statistical data of it is published. The lemma was discussed in the previous section.

The requirement is fulfilled because in the design PFM1 messages contain data for composing the statistical data of resolution types mentioned in the requirement. Additionally, statistical information of pseudonym resolutions which have been informed can be composed by checking which PFM1 messages have corresponding PFM2 messages.

## Countermeasures Against the Adversary Model

### Vehicles

The attack discussed in Section 3.2 where the adversary controls a vehicle and the APRS is unfeasible if there are no available side channels in the APRS. The APRS has a TEE component and in the proposed solution the TEE component enforces that resolution is possible only with a legitimate order from a legitimate judge. Vehicles can notice if the TEE component is faulty by requesting remote attestation from it. However, in the attack, the adversary does not need to make the TEE faulty. It could be sufficient for the adversary to monitor side channels as discussed in Section 6.1. Therefore avoiding side channels in the APRS is sufficient for avoiding the attack.

The attack discussed in Section 3.2 where the adversary could try to send a false ID data from a controlled vehicle to the APRS is not feasible because a vehicle signs its ID before sending it and leakage of private keys is unlikely if on-board units are equipped with a trusted platform module or a TEE (terms hardware security module [29] and trusted component [30] are used also). If a vehicle sends a correctly signed false identity, it will be caught. Signatures are made with private keys and it is assumed that the APRS knows all the legitimate participants of the related identity management system.

The attack discussed in Section 3.2 where the adversary could try to alter data in the pseudonym resolution service by commanding a controlled

vehicle can be mitigated by careful input sanitation if the vehicle can send data to the APRS. The only advantage for the adversary of controlling a vehicle in this case is that it achieves a wider available attack surface.

The attack discussed in Section 7.1 where a vehicle could try to read information for some other vehicle is not possible in the proposed solution and this can be proved with lemma `z_is_not_known_by_the_adversary` on line 995 in the model.

The attack discussed in Section 7.1 where a vehicle could try to alter data in the public forum is not feasible in the proposed solution because the data can be copied to many locations and its integrity can be checked afterwards (requirement integrity protected in Section 5.7 and discussion in Section 6.2)

### Pseudonym Issuing Authority (PIA)

The attack discussed in Section 3.2 where a controlled PIA tries to send false linkage data to the APRS is a real threat against the proposed solution. One possible solution could be to enforce vehicles to send their pseudonyms after issuance to the APRS encrypted for the APRS and signed with long-term keys. If the APRS does not receive matching pseudonym data from the PIA and the vehicle, it can conclude that either the PIA or the vehicle is malicious. It is assumed that the vehicle does not have access to a private key belonging to some other vehicle. The problem is that the PIA and the vehicle can decide not to send any messages to the APRS at all. They will be caught in that case if a non-imported pseudonym is received afterwards by the APRS in a resolution request.

The attack where a controlled PIA does not send any linkage data of some issued pseudonyms to the APRS could be prevented similarly as the previous attack.

The attack where a controlled PIA sends linkage data to the adversary is more likely an issue of a related identity management system.

### Law Enforcement Authority (LEA)

The attack discussed in Section 3.2 where a controlled law enforcement authority representative ties to do unauthorized pseudonym resolution by stealing credentials from an authorized law enforcement authority representative is possible in the solution. A possible solution for the problem is that authorized law enforcement authorities keep track of published PFM1 messages. If a law enforcement authority representative notices a PFM1 message in the public forum with an order ID which belongs to an order belonging to the representative and if the representative was not the one requesting resolution related to the PFM1, then the representative can conclude that its credentials are stolen.

The attack where a controlled LEA representative tries to get unauthorized pseudonym resolution data from the pseudonym resolution service without stolen credentials is not possible without side channels or severe vulnerabilities in an APRS implementation. An order contains an authorized LEA representative ID and the ID is signed by the authorizing judge. The APRS checks the signature and compares the LEA representative ID in the order to the requesting LEA representative ID. This can be proved with lemma `stolen_orders_cannot_be_used_for_res` on line 807 in the model.

The attack where a controlled LEA representative tries to alter data in the APRS by attacking against the APRS is not possible without severe vulnerabilities in an APRS implementation.

The attack discussed in Section 7.1 where an LEA representative tries to infer some extra information from the PF can be prevented by choosing carefully all the pseudonym resolution metadata (for example column $\epsilon_1$ in Table 4).

The attack where a law enforcement authority representative tries to use stolen credentials of some other law enforcement authority representative to ask for permission to resolve from a judge with spoofed justification data and then use the same credentials to send resolution requests to the APRS is possible in the proposed solution if the judge does not notice that the requester is using stolen credentials. The problem can be solved by including a hash $h$(order ID||LEA ID) into variable $\beta_1$ into pseudonym resolution order metadata (Table 2). By adding the hash into pseudonym resolution order metadata the holder of the credentials can notice if its credentials are misused.

### Judge

The attack in Section 7.1 where a judge tries to authorize a law enforcement authority representative to resolve with a fake identity or a stolen identity can be prevented. Authorization with a fake identity does not work, because the APRS checks the signature from the judge. If an order made with stolen credentials is used, the ID of the accountable judge is added to variable $\epsilon_1$ (Table 2). The accountable judge can then notice, that its credentials are used for making the order.

The case where a judge tries to convince that it is not accountable of some resolution permission or resolutions done with the permission can be handled by using a policy where a holder of credentials is accountable no matter who uses the credentials. As discussed before, a judge can notice if its credentials are misused. Misused credentials should be revoked as soon as possible.

The attack where a judge tries to alter published PFMs for getting rid of its accountability is not feasible in the proposed solution because the data

can be copied to many locations and its integrity can be checked afterwards (requirement integrity protected in Section 5.7 and discussion in Section 6.2).

#### Public Forum

The attack where the public forum tries to spoof proof of publishing is not feasible if the public forum implementation option in Section 6.2 is used. The implementation option enforces that at least a proportion of vehicles *can* read a message.

The attack where the public forum tries to alter added values is not possible, because the public forum has the requirement integrity protected and the requirement can be met as discussed in Section 6.2.

#### Accountable Pseudonym Resolution Service (APRS)

The attack discussed in Section 7.1 where the adversary tries to accept unauthorized resolutions by controlling the APRS is not feasible, because there is a TEE-component in the APRS. Allowing only authorized resolutions is enforced with the TEE-component. The TEE-component can be asked to attest that it is correct.

The attack where the adversary tries to avoid publishing PFMs when resolving by controlling the APRS is not feasible because publishing PFMs is enforced with the TEE-component.

The attack where the adversary tries to intercept linkage data import messages by controlling the APRS so that some pseudonyms of a vehicle are not resolvable is not feasible since the channel between the sender and the receiver is assumed to be secure. The sender of data import messages must receive a confirmation from the TEE-component that the data was received.

### 7.3 How vehicles and Judges Can Trust the System

The system must be trustworthy for vehicles and judges. Vehicles do not want to participate a system which leaks their personal information in an uncontrolled way. Judges do not want seem to be accountable for actions which they have not allowed. A vehicle can request the APRS for remote attesting its TEE-component. The TEE in the APRS is the root of trust for vehicles, because it enforces pseudonym resolution to be accountable. As discussed in the previous section, a judge can trust the system as long as it keeps its credentials safe.

### 7.4 Integration to SCMS and PUCA

In this section integration to the two schemes is discussed.

### SCMS

Resolution should be enabled as explained in Section 2.2.6 in Section Pseudonym Resolution. In the pseudonym issuance, issued pseudonyms should be stored somewhere. Then these stored pseudonyms should be resolved after the pseudonym issuance process and sent to the APRS with corresponding linkage data.

### PUCA

In PUCA the Pseudonym Certificate Authority (PCA) *knows* all the pseudonyms but it cannot link these pseudonyms to any long-term identities. Integrating PUCA to the proposed solution is possible in the following way. When the PCA signs pseudonyms to a vehicle it sends these pseudonyms to the APRS. Then the vehicle sends its long-term identity with the issued pseudonyms to the APRS. If the vehicle does not do that, then the APRS can send a misbehaviour report to the Registration Authority (RA) and the pseudonyms will be revoked. The APRS can wait for the message from the vehicle for example for 1 hour.

# 8 Related work

The closest related work is the Conditional Pseudonym Resolution Algorithm In VANETs (CoPRA) [8]. Other related work include Secure revocable anonymous authenticated inter-vehicle communication [20] and V-Tokens for Conditional Pseudonymity in VANETs [35]. The CoPRA will be discussed first and then briefly the other two.

## 8.1 Conditional Pseudonym Resolution Algorithm in VANETs

CoPRA is a proposal for a generic pseudonym resolution protocol in vehicular ad hoc networks which can be integrated into a PKI. In sections 2.2.6 and 2.2.7 V2X pseudonym schemes were discussed. CoPRA is a conditional pseudonym resolution algorithm for such a system.

**The participating entities:** The participating entities in CoPRA are entities from a PKI designed for securing V2X communication [9], such as the root certificate authority, Long Term Certificate Authority (LTCA) and Pseudonym Certificate Authority (PCA). Additionally, there is a Data Protection Agency (DPA) (there can be many of them), which is an optional entity for checking that an authority is authorized to resolve pseudonyms. Authorization information is saved to a certificate issued by a root certificate authority for an authorized authority. For instance, information that an authority is allowed to do pseudonym holder-linkage resolution or pseudonym co-linkage resolution is saved. Additionally, an authority can have validity a time for resolution authorization.

**Pseudonym issuance:** In pseudonym issuance, when a PCA receives a pseudonym certificate request containing a vehicle's long-term identity encrypted for the LTCA and a public key to be certified, the PCA generates a resolution ID $RId_{PC_V}$ by concatenating a hash of the public key and a fresh *random number*. After generating the identifier, the PCA sends to the LTCA a message containing the encrypted long-term identity, the vehicle's signature over the request, a hash of the request and the $RId_{PC_V}$. After receiving the message, the LTCA stores the $RId_{PC_V}$ with the long-term identity and ID of the PCA. Then the LTCA sends a hash of the request, expiration information of the pseudonym being issued and signature over the message back to the PCA. After receiving the message, the PCA stores the identifier of the pseudonym being issued, the resolution ID $RId_{PC_V}$ and identity of the LTCA. Finally the PCA sends the signed pseudonymous certificate back to the vehicle. In the issuance process, both the LTCA and PCA stored escrow data for conditional pseudonym resolution.

**The pseudonym resolution process:** Assume that an authorized authority has a report message from an entity, which is a justification for the resolution. Additionally assume that there is only one pseudonym under resolution and that the authority wants to get holder-linkage (definition 1)

data for the pseudonym. The pseudonym resolution process starts from a point, where a requesting authority sends a resolution request to a DPA with the justification message, a pseudonym, resolution type and a signature over the message (certificate containing authorization data is also sent). The DPA checks that the authority is authorized to proceed with the resolution by checking and verifying the certificate delivered by the authority. If the DPA approves the request, then it sends hash $\delta$ of the justification message and pseudonym, current time, resolution types, and signature over the whole message. In the next protocol step, the authority sends the justification message, the pseudonym, the response from the DPA $res_{DPA}$, resolution types, and signature over the whole message. When the PCA receives the message, it first checks that a DPA approved the resolution. Then it encrypts the resolution ID $RId_{PC_V}$ for the pseudonym, the hash $\delta$ from the DPA and current time for the LTCA. The PCA saved the resolution ID during pseudonym issuance. The PCA sends back to the authority the encrypted data, the same hash $\delta$ which was encrypted, resolution types, the response from the DPA $res_{DPA}$ and signature over the whole message. When the authority receives the response from the PCA, it just forwards the message to the LTCA. The LTCA checks all the signatures and certificates from the authority, the DPA and the PCA. If all checks pass, then the LTCA sends back the hash $\delta$, target vehicles long-term ID, expiry date of the long-term ID (the long-term ID for a vehicle can be changed periodically) and signature over the whole message.

**Accountability in CoPRA:** In this scheme, DPA is made accountable since it approves all the pseudonym resolution requests. Non authorized pseudonym resolution is restricted by organizational separation. However, if the PCA and the LTCA collude, they can resolve pseudonyms without limits. Malicious collusion of the DPA is not needed for malicious pseudonym resolution. Therefore the DPA is not fully accountable. In the conclusion section, the authors of the CoPRA-paper denote that CoPRA could be hardened using a TEE.

**Integrating CoPRA to SCMS:** CoPRA can be integrated to SCMS if the $\text{RA}_{\text{SCMS}}$ is allowed to send an long-term identity of a vehicle to a requester, as discussed in the pseudonym resolution paragraph in Section 2.2.6. The authorized authority can be the misbehaviour authority in SCMS. The DPA can be the $\text{PCA}_{\text{SCMS}}$, because the $\text{PCA}_{\text{SCMS}}$ has the linkage information of the RA-to-PCA hash to a pseudonymous certificate. The $\text{RA}_{\text{SCMS}}$ should approve a request from the misbehaviour authority in SCMS only if the request is approved by the $\text{PCA}_{\text{SCMS}}$.

## 8.2 Secure revocable anonymous authenticated inter-vehicle communication

Secure revocable anonymous authenticated inter-vehicle communication is a pseudonym scheme with pseudonym resolution capabilities. Pseudonym resolution is possible by co-operation of preset number of authorities and co-operation is enforced by using cryptographic methods.

Authors developed the scheme focusing on reducing the danger of misuse by a single compromised authority and enforcing the many-eye principle for resolving identities. Authors also set a broad requirement for their scheme which states that by using their scheme no more personal information should be possible to infer than without using their scheme. They use pseudonyms to achieve unlinkability. They use magic-ink signatures to achieve the objective of many-eye identity resolution.

For each vehicle using the scheme there is a *tag*. The system knows the linkage between vehicles and tags. In identity resolution a tag can be resolved and therefore the corresponding vehicle. To resolve the tag, preset number of authorities should co-operate. The many-eye principle is met in that way.

There is no accountable authorities for a pseudonym resolution in the scheme. If all the authorities needed for resolution collude, users of the system will not know about that. Being not accountable is a difference to the design presented in this thesis. In the scheme, linkage data is not stored in a single place. Linkage data between tags and vehicles is, but leaking tags is not sufficient to link pseudonyms to long term identities. In the presented design, linkage data is stored possibly in one place.

The scheme does not consider the granularity of pseudonym resolution. No statistical information is available of pseudonym resolutions.

## 8.3 V-Tokens for Conditional Pseudonymity in VANETs

V-Tokens for Conditional Pseudonymity in VANETs is a solution for mitigating the risk of storing linkage data to a same place. By using V-Tokens, resolution information is stored into pseudonyms. Resolution information can be used only by many authorized co-operating authorities.

A V-token is in practice encrypted data produced with a secret-sharing scheme. A token is encrypted with a public key of resolution authorities. To decrypt the key, a minimum proportion of resolution authorities should co-operate.

There is no accountable authorities for a pseudonym resolution in this protocol either. The scheme is not accountable. Being not accountable is a difference to the design presented in this thesis. In this scheme, linkage data is not stored in a same place. That is a difference to the design presented in this thesis.

This scheme does not consider the granularity of pseudonym resolution either. And no statistical information is available of pseudonym resolutions.

# 9 Conclusion

The answer to the main research question "*How can we achieve accountable pseudonym resolution in V2X communication systems?*" follows.

Accountable resolution can be achieved by making some entity accountable of resolution. In the proposed solution the accountable entity is judge. In the solution, judges authorize resolutions by giving resolution orders. Pseudonym resolution is possible only with a legitimate order. If an order is used, metadata of the usage is published to the public forum. In the solution the public forum is responsible for making all the published messages available to *everyone*. Therefore judges are made accountable by publishing metadata of resolutions which are authorized by the judges.

As discussed in Section 7.2, to achieve accountable resolution the system must be secure. Otherwise, pseudonym resolution is possible without an order from an accountable judge. The most critical part of the system from the security point of view is the APRS and its TEE-component. Linkage data should be imported to the APRS in a secure way and linkage data should be stored by the APRS in a secure way. If data import and data storage are secure, linkage data cannot be read in an unauthorized way. The TEE-component can be used in secure import (encrypt for the component) and in secure storage (sealing by the component). As discussed in Section 6.1, there must not be side channels available in the APRS. The APRS must be implemented in a way it tolerates intrusions without leaking linkage data and losing accountability of the system. The TEE-component is an important part of the intrusion tolerance.

The mechanism for making judges accountable in the proposed solution is to enforce publishing messages to the public forum by the APRS when a resolution happens. The public forum is a central component of the accountability mechanism. It is possible to make sure that a proportion of all the vehicles *can* read a published message, as explained in Section 6.2.

# References


[1] *ETSI TS 102 731 V1.1.1 (2010–09)*, September 2010. http://www.etsi.org/deliver/etsi_ts/102700_102799/102731/01.01.01_60/ts_102731v010101p.pdf.

[2] *ETSI TS 102 940 V1.1.1 (2012–06)*, June 2012. http://www.etsi.org/deliver/etsi_ts/102900_102999/102940/01.01.01_60/ts_102940v010101p.pdf.

[3] *1609.2-2016 - IEEE Standard for Wireless Access in Vehicular Environments–Security Services for Applications and Management Messages*, 2016. https://standards.ieee.org/findstds/standard/1609.2-2016.html.

[4] *ETSI TR 102 893 V1.2.1 (2017–03)*, March 2017. http://www.etsi.org/deliver/etsi_tr/102800_102899/102893/01.02.01_60/tr_102893v010201p.pdf.

[5] Armknecht, F., Festag, A., Westhoff, D., and Zeng, K.: *Cross-layer Privacy Enhancement and Non-repudiation in Vehicular Communication.* In *Communication in Distributed Systems - 15. ITG/GI Symposium*, pages 1–12, February 2007.

[6] Asokan, N., Davi, Lucas, Dmitrienko, Alexandra, Heuser, Stephan, Kostiainen, Kari, Reshetova, Elena, and Sadeghi, Ahmad Reza: *Mobile Platform Security.* Morgan & Claypool, 2013, ISBN 9781627050982.

[7] Basin, David, Cremers, Cas, Dreier, Jannik, Meier, Simon, Sasse, Ralf, and Schmidt, Benedikt: *Tamarin prover*, 2017. https://tamarin-prover.github.io, visited on 2017-11-15.

[8] Bißmeyer, N., Petit, J., and Bayarou, K. M.: *Copra: Conditional pseudonym resolution algorithm in VANETs.* In *2013 10th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*, pages 9–16, March 2013.

[9] Bißmeyer, Norbert, Stbing, Hagen, Schoch, Elmar, and Lonc, Brigitte: *A Generic Public Key Infrastructure for Securing Car-to-X Communication.* In *18th World Congress on Intelligent Transport Systems.* https://www.researchgate.net/publication/268100474_A_Generic_Public_Key_Infrastructure_for_Securing_Car-to-X_Communication, visited on 2017-09-17.

[10] Brasser, Ferdinand, Müller, Urs, Dmitrienko, Alexandra, Kostiainen, Kari, Capkun, Srdjan, and Sadeghi, Ahmad Reza: *Software Grand Exposure: SGX Cache Attacks Are Practical.* arXiv:1702.07521 [cs], Febru-

ary 2017. http://arxiv.org/abs/1702.07521, visited on 2017-10-23, arXiv: 1702.07521.

[11] Chaum, David and Heyst, Eugène van: *Group Signatures*. In *Advances in Cryptology — EUROCRYPT '91*, Lecture Notes in Computer Science, pages 257–265. Springer, Berlin, Heidelberg, April 1991, ISBN 978-3-540-54620-7 978-3-540-46416-7. https://link.springer.com/chapter/10.1007/3-540-46416-6_22, visited on 2017-09-16.

[12] Choi, Jong Youl, Jakobsson, Markus, and Wetzel, Susanne: *Balancing Auditability and Privacy in Vehicular Networks*. In *Proceedings of the 1st ACM International Workshop on Quality of Service & Security in Wireless and Mobile Networks*, Q2SWinet '05, pages 79–87, New York, NY, USA, 2005. ACM, ISBN 978-1-59593-241-9. http://doi.acm.org/10.1145/1089761.1089775, visited on 2017-09-14.

[13] Dolev, D. and Yao, A. C.: *On the security of public key protocols*. In *22nd Annual Symposium on Foundations of Computer Science (sfcs 1981)*, pages 350–357, October 1981.

[14] Douceur, John R.: *The Sybil Attack*. In *Peer-to-Peer Systems*, Lecture Notes in Computer Science, pages 251–260. Springer, Berlin, Heidelberg, March 2002, ISBN 978-3-540-44179-3 978-3-540-45748-0. https://link.springer.com/chapter/10.1007/3-540-45748-8_24, visited on 2017-11-19.

[15] Du, S., Ibrahim, M., Shehata, M., and Badawy, W.: *Automatic License Plate Recognition (ALPR): A State-of-the-Art Review*. IEEE Transactions on Circuits and Systems for Video Technology, 23(2):311–325, February 2013, ISSN 1051-8215.

[16] Ekberg, J. E., Kostiainen, K., and Asokan, N.: *The Untapped Potential of Trusted Execution Environments on Mobile Devices*. IEEE Security Privacy, 12(4):29–37, July 2014, ISSN 1540-7993.

[17] European Parliament and the Council of 7 July 2010: *Directive 2010/40/EU*. http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:32010L0040, visited on 2017-11-23.

[18] Eze, E. C., Zhang, S., and Liu, E.: *Vehicular ad hoc networks (VANETs): Current state, challenges, potentials and way forward*. In *2014 20th International Conference on Automation and Computing*, pages 176–181, September 2014.

[19] Fan, Li, Cao, Pei, Almeida, Jussara, and Broder, Andrei Z.: *Summary Cache: A Scalable Wide-area Web Cache Sharing Protocol*. IEEE/ACM Trans. Netw., 8(3):281–293, June 2000, ISSN 1063-6692. http://dx.doi.org/10.1109/90.851975, visited on 2017-11-13.

[20] Fischer, Lars, Aijaz, Amer, Eckert, Claudia, and Vogt, David: *Secure revocable anonymous authenticated inter-vehicle communication (SRAAC)*. 12.

[21] Förster, David, Kargl, Frank, and Löhr, Hans: *PUCA: A pseudonym scheme with strong privacy guarantees for vehicular ad-hoc networks*. Ad Hoc Networks, 37, Part 1:122–132, February 2016, ISSN 1570-8705. `http://www.sciencedirect.com/science/article/pii/S1570870515002280`.

[22] Goldreich, Oded, Goldwasser, Shafi, and Micali, Silvio: *How to Construct Random Functions*. J. ACM, 33(4):792–807, August 1986, ISSN 0004-5411. `http://doi.acm.org/10.1145/6490.6503`, visited on 2017-11-22.

[23] Goldreich, Oded and Ostrovsky, Rafail: *Software Protection and Simulation on Oblivious RAMs*. J. ACM, 43(3):431–473, May 1996, ISSN 0004-5411. `http://doi.acm.org/10.1145/233551.233553`, visited on 2017-11-30.

[24] Guo, J., Baugh, J. P., and Wang, S.: *A Group Signature Based Secure and Privacy-Preserving Vehicular Communication Framework*. In *2007 Mobile Networking for Vehicular Environments*, pages 103–108, May 2007.

[25] Kamat, Pandurang, Baliga, Arati, and Trappe, Wade: *An Identity-based Security Framework For VANETs*. In *Proceedings of the 3rd International Workshop on Vehicular Ad Hoc Networks*, VANET '06, pages 94–95, New York, NY, USA, 2006. ACM, ISBN 978-1-59593-540-3. `http://doi.acm.org/10.1145/1161064.1161083`, visited on 2017-09-16.

[26] Kargl, F., Papadimitratos, P., Buttyan, L., Müter, M., Schoch, E., Wiedersheim, B., Thong, T. V., Calandriello, G., Held, A., Kung, A., and Hubaux, J. P.: *Secure vehicular communication systems: implementation, performance, and research challenges*. IEEE Communications Magazine, 46(11):110–118, November 2008, ISSN 0163-6804.

[27] L'Assemblée fédérale de la Confédération suisse: *Loi fédérale sur le renseignement*, September 2015. `https://www.admin.ch/opc/fr/federal-gazette/2015/6597.pdf`, visited on 2017-10-10.

[28] McKeen, Frank, Alexandrovich, Ilya, Berenzon, Alex, Rozas, Carlos V., Shafi, Hisham, Shanbhogue, Vedvyas, and Savagaonkar, Uday R.: *Innovative Instructions and Software Model for Isolated Execution*. In *Proceedings of the 2Nd International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '13, pages 10:1–10:1,

New York, NY, USA, 2013. ACM, ISBN 978-1-4503-2118-1. http://doi.acm.org/10.1145/2487726.2488368.

[29] Papadimitratos, P., Buttyan, L., Holczer, T., Schoch, E., Freudiger, J., Raya, M., Ma, Z., Kargl, F., Kung, A., and Hubaux, J. P.: *Secure vehicular communication systems: design and architecture*. IEEE Communications Magazine, 46(11):100–109, November 2008, ISSN 0163-6804.

[30] Papadimitratos, P., Buttyan, L., Hubaux, J. P., Kargl, F., Kung, A., and Raya, M.: *Architecture for Secure and Private Vehicular Communications*. In *2007 7th International Conference on ITS Telecommunications*, pages 1–6, June 2007.

[31] Petit, J., Schaub, F., Feiri, M., and Kargl, F.: *Pseudonym Schemes in Vehicular Networks: A Survey*. IEEE Communications Surveys Tutorials, 17(1):228–255, 2015, ISSN 1553-877X.

[32] Pfitzmann, Andreas and Hansen, Marit: *A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management*, August 2010. http://dud.inf.tu-dresden.de/Anon_Terminology.shtml, visited on 2017-11-20, Version v0.34.

[33] Raya, Maxim and Hubaux, Jean Pierre: *The Security of Vehicular Ad Hoc Networks*. In *Proceedings of the 3rd ACM Workshop on Security of Ad Hoc and Sensor Networks*, SASN '05, pages 11–21, New York, NY, USA, 2005. ACM, ISBN 978-1-59593-227-3. http://doi.acm.org/10.1145/1102219.1102223, visited on 2017-11-23.

[34] Raya, Maxim and Hubaux, Jean Pierre: *Securing Vehicular Ad Hoc Networks*. J. Comput. Secur., 15(1):39–68, January 2007, ISSN 0926-227X. http://dl.acm.org/citation.cfm?id=1370616.1370618, visited on 2017-10-23.

[35] Schaub, F., Kargl, F., Ma, Z., and Weber, M.: *V-Tokens for Conditional Pseudonymity in VANETs*. In *2010 IEEE Wireless Communication and Networking Conference*, pages 1–6, April 2010.

[36] Schaub, F., Ma, Z., and Kargl, F.: *Privacy Requirements in Vehicular Communication Systems*. In *2009 International Conference on Computational Science and Engineering*, volume 3, pages 139–145, August 2009.

[37] Schmidt, B., Meier, S., Cremers, C., and Basin, D.: *Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties*. In *2012 IEEE 25th Computer Security Foundations Symposium*, pages 78–94, June 2012.

[38] Shamir, Adi: *Identity-Based Cryptosystems and Signature Schemes*. In *Advances in Cryptology*, Lecture Notes in Computer Science, pages 47–53. Springer, Berlin, Heidelberg, August 1984, ISBN 978-3-540-15658-1 978-3-540-39568-3. https://link.springer.com/chapter/10.1007/3-540-39568-7_5, visited on 2017-09-16.

[39] Silvennoinen, Aku: *Accountable pseudonym resolution service Tamarin model*, 2017. http://iki.fi/aku.silvennoinen/v2x/accountable_pseudonym_resolution_service.spthy, visited on 2017-12-1.

[40] Silvennoinen, Aku: *Accountable pseudonym resolution service Tamarin proof*, 2017. http://iki.fi/aku.silvennoinen/v2x/accountable_pseudonym_resolution_service_proof.txt, visited on 2017-12-1.

[41] Stevenson, Angus: *Oxford Dictionary of English*. ISBN 9780199571123. http://www.oxfordreference.com/view/10.1093/acref/9780199571123.001.0001/acref-9780199571123.

[42] Tamrakar, Sandeep, Liu, Jian, Paverd, Andrew, Ekberg, Jan Erik, Pinkas, Benny, and Asokan, N.: *The Circle Game: Scalable Private Membership Test Using Trusted Hardware*. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pages 31–44, New York, NY, USA, 2017. ACM, ISBN 978-1-4503-4944-4. http://doi.acm.org/10.1145/3052973.3053006, visited on 2017-11-30.

[43] Whyte, W., Weimerskirch, A., Kumar, V., and Hehn, T.: *A security credential management system for V2v communications*. In *2013 IEEE Vehicular Networking Conference*, pages 1–8, December 2013.

[44] Xu, Y., Cui, W., and Peinado, M.: *Controlled-Channel Attacks: Deterministic Side Channels for Untrusted Operating Systems*. In *2015 IEEE Symposium on Security and Privacy*, pages 640–656, May 2015.

[45] Zeng, Ke: *Pseudonymous PKI for Ubiquitous Computing*. In *Public Key Infrastructure*, Lecture Notes in Computer Science, pages 207–222. Springer, Berlin, Heidelberg, June 2006, ISBN 978-3-540-35151-1 978-3-540-35152-8. https://link.springer.com/chapter/10.1007/11774716_17, visited on 2017-09-16.

# A Tamarin Prover Model

Listing 1: Tamarin prover model

```
theory accountable_pseudonym_resolution_service
begin

/*******************************************************
*   ==================================================
*
*   Accountable Pseudonym Resolution Service
*
*   Model and Lemmas
*
*   ==================================================
*
*   Author: Aku Silvennoinen
*
*   For Tamarin prover version: 1.2.2
*
*******************************************************/

functions: h1/1, h2/1
builtins: asymmetric-encryption, signing, symmetric-encryption

/*******************************************************
*   Restrictions
*******************************************************/

restriction Unique:
"
  All x #i #j. UniqueFact(x) @ #i & UniqueFact(x) @ #j ==> #i = #j
"

restriction Equality:
"
  All x y #i. Eq(x,y) @ #i ==> x = y
"

/*******************************************************
*   Abbreviations and action fact naming conventions
*******************************************************/
/*

  S -> Secure channel

  PFM1 -> First public forum message
  PFM2 -> Second public forum message

  Action facts denoting receiving or sending messages are named following the next convention:
  participant1_'sen'|'rec'_messageID_'to'|'from'_participant2()

  sen -> sends
  rec -> receives

  DI -> Data import
  GRS -> Generic resolution system

  a -> alpha
  b -> beta
  g -> gamma
  d -> delta
  e -> epsilon
  z -> zeta
  eta -> eta

  sig -> signature
  res -> resolution

  ord -> order
  pub -> public

*/

/*******************************************************
*   1. Rules
*******************************************************/
/*******************************************************
*   1.1 PKI
*******************************************************/

```

```
78  // Each pub key can be read by the adversary
79  rule Get_pk:
80    [ !Pk(A,pubkey) ]
81  --[ Adversary_knows_pub_key(pubkey) ]->
82    [ Out(pubkey) ]
83
84  rule Get_pkV:
85    [ !PkV(A,pubkey) ]
86  --[ Adversary_knows_pub_key(pubkey) ]->
87    [ Out(pubkey) ]
88
89  rule Register_pk_V_lt:
90    [ Fr(~ltk) ]
91  -->
92    [ !LtkV($V,~ltk),!PkV($V,pk(~ltk)) ]
93
94  // Judge
95  rule Register_pk_J:
96    [ Fr(~ltk) ]
97  --[ Register_sk_J(~ltk)
98    , UniqueFact('register_pk_J')
99    ]->
100   [ !Ltk('J',~ltk),!Pk('J',pk(~ltk)) ]
101
102 // APRS
103 rule Register_pk_APRS:
104   [ Fr(~ltk) ]
105 --[ UniqueFact('register_pk_APRS')]->
106   [ !Ltk('APRS',~ltk),!Pk('APRS',pk(~ltk)) ]
107
108 // LEA_1
109 rule Register_pk_LEA:
110   [ Fr(~ltk) ]
111 --[ Register_sk_LEA(~ltk)
112   , Register_pk_LEA(pk(~ltk))
113   , UniqueFact('register_pk_LEA')
114   ]->
115   [ !Ltk('LEA',~ltk),!Pk('LEA',pk(~ltk)) ]
116
117 // Malicious LEA
118 rule Register_pk_malicious_LEA:
119   [ Fr(~ltk) ]
120 --[ UniqueFact('register_pk_malicious_LEA')]->
121   [ !Ltk('MLEA',~ltk),!Pk('MLEA',pk(~ltk)) ]
122
123 /*******************************************************
124 *   1.2 Channel rules
125 *******************************************************/
126 /*******************************************************
127 *   1.2.1 Secure channel
128 *******************************************************/
129
130 rule Chan_Out_S:
131   [ Out_S($A,$B,x) ]
132 --[ Chan_Out_S($A,$B,x) ]->
133   [ !S($A,$B,x) ]
134
135 rule Chan_In_S:
136   [ !S($A,$B,x) ]
137 --[ Chan_In_S($A,$B,x) ]->
138   [ In_S($A,$B,x) ]
139
140 /*******************************************************
141 *   1.3 CID delivery
142 *******************************************************/
143
144 rule APRS_1:
145   [ Fr(~cid)
146   , !PkV($V,pkV_lt)
147   ]
148 --[ APRS_sen_CIDD1_to_CIDS(~cid,pkV_lt)
149   , UniqueFact(<'APRS_5',pkV_lt>)  // generate only one cid for a vehicle
150   ]->
151   [ Out_S('APRS','CIDS',<h1(pkV_lt),aenc(~cid,pkV_lt)>)
152   , !APRS_1(~cid,pkV_lt) // aprs stores the cid and the public key for later use
153   ]
154
155 rule CIDS_1:
156   [ In_S('APRS','CIDS',<idhash,enc_cid>) ]
157 --[ CIDS_rec_CIDD1_from_APRS(idhash,enc_cid)
158   , UniqueFact(<'APRS_5',idhash,enc_cid>) // store a cid only once
159   ]->
160   [ !CIDS(idhash,enc_cid)
161   , Out(<idhash,enc_cid>)
```

```
  ]

rule V_1:
  [ !LtkV($V,skV_lt)
  , !CIDS(idhash,enc_cid)
  ]
--[ Vehicle_finds_cid_and_stores_it(adec(enc_cid,skV_lt),pk(skV_lt))
  , Eq(idhash,h1(pk(skV_lt)))
  , UniqueFact(<'V_1',skV_lt>) // no need to store a cid twice
  ]->
  [ !V_1(pk(skV_lt),adec(enc_cid,skV_lt)) ]

/*******************************************************
*   1.4 Data import
*******************************************************/

rule PIA_1:
  [ Fr(~pseu)            // generate speudonym
  , !PkV($V,pkV_lt)      // lookup pub-key of V
  ]
--[ PIA_sen_DI_to_APRS(~pseu,pkV_lt) ]->
  [ Out_S('PIA','APRS',<~pseu,pkV_lt,'DI'>)
  , !LEA(~pseu)          // LEA acquiring the pseudonym
  ]

rule APRS_2:
  [ In_S('PIA','APRS',<pseu,pkV_lt,'DI'>) ]
--[ APRS_rec_DI_from_PIA(pseu,pkV_lt) ]->
  [ !APRS_2(pseu,pkV_lt) ]

/*******************************************************
*   1.5 Generic resolution system
*******************************************************/

rule LEA_1:
  [ Fr(~a)          // variable for resolution permission request
  , !Pk('LEA',pkLEA1) // lookup pub-key of LEA
  ]
--[ LEA_sen_GRS1_to_J(~a)
  ]->
  [ Out_S('LEA','J',<~a,pkLEA1,'GRS1'>)
  , !LEA_1(~a)
  ]

// g contains LEA ID. That is modeled by delivering LEAs pub key with g. J signs the
    ↪ pub key also.
rule J_1:
  let sig=sign(<<~b1,~ordID>,~b2,<~g,pkLEA>>,skJ)
  in
  [ In_S('LEA','J',<a,pkLEA,'GRS1'>)
  , !Ltk('J',skJ) // lookup private key of J
  , Fr(~b1)       // variable for resolution order meta data
  , Fr(~ordID)    // fresh order ID (same for b1, b2 and b3)
  , Fr(~b2)       // variable for delayed resolution order meta data
  , Fr(~g)        // variable for resolution order
  ]
--[ J_rec_GRS1_from_LEA(a,pkLEA)
  , J_sen_GRS2_to_LEA(a,<~b1,~ordID>,~b2,<~g,pkLEA>,sig)
  ]->
  [ Out_S('J','LEA',<a,<~b1,~ordID>,~b2,<~g,pkLEA>,sig,'GRS2'>)
  , !J_1(~ordID)
  ]

rule LEA_2:
  [ In_S('J','LEA',<a,<b1,ordID>,b2,<g,pkLEA>,sig,'GRS2'>)
  , !LEA_1(a)
  ]
--[ LEA_rec_GRS2_from_J(a,<b1,ordID>,b2,<g,pkLEA>,sig)
  , UniqueFact(<'LEA_2',a,ordID>)
  ]->
  [ !LEA_2_order(<<b1,ordID>,b2,<g,pkLEA>,sig>) ]

rule LEA_3:
  [ !LEA_2_order(<<b1,ordID>,b2,<g,pkLEA>,sig>)
  , Fr(~d)              // variable for resolution request
  , Fr(~resID)          // d contains resolution ID
  , !Ltk('LEA',skLEA)   // lookup private key of LEA
  , !LEA(pseu)          // d contains a pseudonym as an argument
  ]
--[ LEA_sen_GRS3_to_APRS(<b1,ordID>,b2,<g,pkLEA>,<~d,~resID,pseu>,sig,sign(pk(skLEA),
    ↪ skLEA)) ]->
  [ Out_S('LEA','APRS',<<b1,ordID>,b2,<g,pkLEA>,<~d,~resID,pseu>,sig,sign(pk(skLEA),
      ↪ skLEA),'GRS3'>)
  , LEA_3(<~d,~resID>)
```

```
  , !LEA_3_res_request(<<b1,ordID>,b2,<g,pkLEA>,<~d,~resID,pseu>,sig,sign(pk(skLEA),
      ↪ skLEA),'GRS3'>)
  ]

// Protocol proceeds only if Js sig is correct and if the sender is authorized to
    ↪ resolve.
rule APRS_3:
  let sig_2=sign(<<b1,ordID>,<~e1,~pub_resID>,h1(<b2,~e2,aenc(~k,pkV_lt),senc(~z,~k),
      ↪ h2(<cid,~s>)>),'GRS4'>,skAPRS)
      PFM1=<<<b1,ordID>,<~e1,~pub_resID>,h1(<b2,~e2,aenc(~k,pkV_lt),senc(~z,~k),h2(<
          ↪ cid,~s>)>),'GRS4'>,sig_2>
  in
  [ In_S('LEA','APRS',<<b1,ordID>,b2,<g,pkLEA_2>,<d,resID,pseu>,sig_1,sig_3,'GRS3'>)
  , !Ltk('APRS',skAPRS)    // lookup private-key of APRS
  , !Pk('J',pkJ)           // lookup pub-key of J
  , !APRS_2(pseu,pkV_lt)   // lookup identity-pseudonym pair of a vehicle
  , !APRS_1(cid,pkV_lt)    // lookup CID
  , Fr(~k)                 // symmetric key for the PFM1
  , Fr(~s)                 // salt for obfuscating the hash
  , Fr(~e1)                // variable for resolution meta data
  , Fr(~e2)                // variable for delayed resolution meta data
  , Fr(~z)                 // variable for target informing data
  , Fr(~pub_resID)         // resolution id for APRS for keeping track of relations of
      ↪ PFM1s and resolution requests
  ]
--[ APRS_rec_GRS3_from_LEA(<<b1,ordID>,b2,<g,pkLEA_2>,<d,resID,pseu>,sig_1,sig_3,'GRS3
    ↪ '>)
  , APRS_sen_GRS4_to_PF(<<b1,ordID>,<~e1,~pub_resID>,<b2,~e2,~k,pkV_lt,~z,~k,cid,~s>>,
      ↪ sig_2)
  , Eq(verify(sig_1,<<b1,ordID>,b2,<g,pkLEA_2>>,pkJ),true)
  , Eq(verify(sig_3,pkLEA_2,pkLEA_2),true)
  , UniqueFact(<'APRS_3_ordID_resID',ordID,resID>) // avoid replay attacs
  ]->
  [ Out(PFM1)
  , !APRS_3_PFM1_data(<b1,ordID>,b2,<d,resID>,<~e1,~pub_resID>,~e2,~s,~k,~z,cid,pkV_lt
      ↪ ,pseu) // used for keeping state in APRS
  ]

rule PF_1:
  let PFM1=<<<b1,ordID>,<e1,pub_resID>,h1(<b2,e2,aenc(k,pkV_lt),senc(z,k),h2(<cid,s>)
      ↪ >),'GRS4'>,sig_2>
  in
  [ In(PFM1)
  , !Pk('APRS',pkAPRS)  // lookup pub-key of APRS
  ]
--[ PF_rec_GRS4_from_APRS(<b1,ordID>,<e1,pub_resID>)
  , Eq(verify(sig_2,<<b1,ordID>,<e1,pub_resID>,h1(<b2,e2,aenc(k,pkV_lt),senc(z,k),h2(<
      ↪ cid,s>)>),'GRS4'>,pkAPRS),true)
  , UniqueFact(<'PF_1',pub_resID>) ]->
  [ !PF_storage(PFM1) ]

rule APRS_4:
  let PFM1=<<<b1,ordID>,<e1,pub_resID>,h1(<b2,e2,aenc(k,pkV_lt),senc(z,k),h2(<cid,s>)
      ↪ >),'GRS4'>,sig_2>
  in
  [ !PF_storage(PFM1)
  , !APRS_3_PFM1_data(<b1,ordID>,b2,<d,resID>,<e1,pub_resID>,e2,s,k,z,cid,pkV_lt,pseu)
      ↪  // APRS state related to resID is restored here
  , !Ltk('APRS',skAPRS) // lookup private-key of APRS
  , Fr(~eta)            // variable for resolution result (for LEA)
  ]
--[ APRS_sen_GRS6_to_LEA(<d,resID,pseu>,<~eta,pkV_lt>)
  , Eq(verify(sig_2,<<b1,ordID>,<e1,pub_resID>,h1(<b2,e2,aenc(k,pkV_lt),senc(z,k),h2(<
      ↪ cid,s>)>),'GRS4'>,pk(skAPRS)),true)
  ]->
  [ Out_S('APRS','LEA',<sign(<<d,resID,pseu>,~eta>,skAPRS),<<d,resID,pseu>,<~eta,pkV_
      ↪ lt>>,'GRS6'>)
  , Out_S('APRS','J',<ordID,resID,'GRS6_2'>)     // notification for J (it can then
      ↪ send an informing order)
  ]

rule LEA_4:
  [ In_S('APRS','LEA',<sig,<<d,resID,pseu>,<eta,pkV_lt>>,'GRS6'>)
  , LEA_3(<d,resID>)
  , !Pk('APRS',pkAPRS)      // lookup pub-key of APRS
  ]
--[ LEA_rec_GRS6_from_APRS(<d,resID,pseu>,<eta,pkV_lt>)
  , Eq(verify(sig,<<d,resID,pseu>,eta>,pkAPRS),true)
  ]->
  [ !LEA_4(d,<eta,pkV_lt>) ]

rule J_2:
  [ In_S('APRS','J',<ordID,resID,'GRS6_2'>)
  , !J_1(ordID)
```

```
  ]
--[ J_receives_information_of_res(ordID,resID)
  , UniqueFact(<'J_2',ordID,resID>)
  ]->
  [ J_2(ordID,resID) ]

rule J_3:
  [ J_2(ordID,resID)
  , Fr(~b3)     // variable for delayed additional resolution order meta data
  ]
--[ J_sen_GRS7_to_APRS(~b3,ordID,resID) ]->
  [ Out_S('J','APRS',<<~b3,ordID>,resID,'GRS7'>) ]

rule APRS_5:
  let sig=sign(<<b3,ordID>,~e3,b2,<e2,pub_resID>,aenc(k,pkV_lt),senc(z,k),h2(<cid,s>),s>,skAPRS)
      PFM2=<<b3,ordID>,~e3,b2,<e2,pub_resID>,aenc(k,pkV_lt),senc(z,k),h2(<cid,s>),s,sig,'GRS8'>
  in
  [ In_S('J','APRS',<<b3,ordID>,resID,'GRS7'>)
  , !APRS_3_PFM1_data(<b1,ordID>,b2,<d,resID>,<e1,pub_resID>,e2,s,k,z,cid,pkV_lt,pseu)
      // APRS state related to b1 is restored here
  , Fr(~e3)                  // variable for delayed additional resolution meta data
  , !Ltk('APRS', skAPRS)  // lookup private-key of APRS
  ]
--[ APRS_rec_GRS7_from_J(<b3,ordID>,resID)
  , APRS_sen_GRS8_to_PF(<b3,ordID>,~e3,b2,<e2,pub_resID>,pkV_lt,z,k,cid,s,sig)
  , UniqueFact(<'APRS_4',resID>)
  ]->
  [ Out(PFM2) ]

rule PF_2:
  let sig=sign(<<b3,ordID>,e3,b2,<e2,pub_resID>,aenc(k,pkV_lt),senc(z,k),h2(<cid,s>),s>,skAPRS)
      PFM2=<<b3,ordID>,e3,b2,<e2,pub_resID>,aenc(k,pkV_lt),senc(z,k),h2(<cid,s>),s,sig,'GRS8'>
  in
  [ In(PFM2)
  , !Pk('APRS',pkAPRS)      // lookup pub-key of APRS
  ]
--[ PF_rec_GRS8_from_APRS(<b3,ordID>,pub_resID)
  , Eq(verify(sig,<<b3,ordID>,e3,b2,<e2,pub_resID>,aenc(k,pkV_lt),senc(z,k),h2(<cid,s>),s>,pkAPRS),true)
  , UniqueFact(<'PF_2',pub_resID>)
  ]->
  [ !PF_storage(PFM2) ]

rule V_2:
  let PFM2=<<b3,ordID>,e3,b2,<e2,pub_resID>,enc_k,enc_z,hash_cid,s,sig,'GRS8'>
      PFM1=<<<b1,ordID>,<e1,pub_resID>,hash,'GRS4'>,sig_2>
  in
  [ !V_1(pkV_lt,cid)
  , !PF_storage(PFM1)
  , !PF_storage(PFM2)
  , !LtkV($V,skV_lt)
  ]
--[ Vehicle_finds_target_data(pkV_lt,sdec(enc_z,adec(enc_k,skV_lt)))
  , Eq(hash,h1(<b2,e2,enc_k,enc_z,hash_cid>))
  , Eq(hash_cid,h2(<cid,s>))
  , Eq(pk(skV_lt),pkV_lt)
  ]->
  [ !V_2(pkV_lt,sdec(enc_z,adec(enc_k,skV_lt))) ]

/*****************************************************
*   1.7 Malicious entities
*****************************************************/
/*****************************************************
*   1.7.1 LEA steals an order and tries to use it to
*         resolve
*
*         This rule is relevant because in the model
*         orders from J are not leaked, but malicious
*         LEA entities should be considered (an LEA steals
*         an order from another LEA).
*****************************************************/

rule LEA_5:
  [ !LEA_2_order(<<b1,ordID>,b2,<g,pkLEA>,sign(<<b1,ordID>,b2,<g,pkLEA>>,skJ)>) ]
--[ LEA_leaks_order(<<b1,ordID>,b2,<g,pkLEA>,skJ>) ]->
  [ Out(<<b1,ordID>,b2,<g,pkLEA>,sign(<<b1,ordID>,b2,<g,pkLEA>>,skJ)>) ]

rule LEA_6:
  [ !LEA_3_res_request(<<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sign(<<b1,ordID>,b2,<g,pkLEA>>,skJ),sign(pk(skLEA),skLEA),'GRS3'>) ]
```

```
--[ LEA_leaks_request(<<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,skJ,skLEA>) ]->
  [ Out(<<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sign(<<b1,ordID>,b2,<g,pkLEA>>,skJ),
      ↪ sign(pk(skLEA),skLEA),'GRS3'>) ]

rule Malicious_LEA_1:
  [ !Ltk('MLEA',skMLEA) // lookup private key of MLEA
  ]
--[ Malicious_LEA_leaks_its_private_key() ]->
  [ Out(skMLEA) ]

rule Malicious_LEA_2:
  [ In(ad_data) ]
--[ Malicious_LEA_sends_res_request_to_APRS_with_data_from_adversary(ad_data) ]->
  [ Out_S('LEA','APRS',ad_data) ]

/*************************************************************
*   2. Lemmas
*************************************************************/
/*************************************************************
*   2.1 Sources lemmas
*************************************************************/

lemma origin_of_b1_ordID_and_b2 [sources]:
"
  All b1 ordID e1 pub_resID b2 e2 z sig_2 k s cid pkV_lt #i.
      APRS_sen_GRS4_to_PF(<<b1,ordID>,<e1,pub_resID>,<b2,e2,k,pkV_lt,z,k,cid,s>>,sig_
          ↪ 2) @ #i
  ==>
      ( Ex a g sig #j.
          J_sen_GRS2_to_LEA(a,<b1,ordID>,b2,g,sig) @ #j
      &   j<i
      )
"

lemma origin_of_b1_ordID_b2_g_pkLEA_skJ_and_skLEA [sources]:
"
  All b1 ordID b2 g pkLEA d resID pseu skJ skLEA #i.
     LEA_leaks_request(<<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,skJ,skLEA>) @ #i
  ==>
      ( Ex a sig #j #k #l #m.
          J_sen_GRS2_to_LEA(a,<b1,ordID>,b2,<g,pkLEA>,sig) @ #j
      &   Register_sk_J(skJ) @ #k
      &   Register_sk_LEA(skLEA) @ #l
      &   Register_pk_LEA(pkLEA) @ #m
      &   j<i
      &   k<i
      &   l<i
      &   m<i
      )
"

lemma origin_of_b1_ordID_b2_g_pkLEA_skJ [sources]:
"
  All b1 ordID b2 g pkLEA skJ #i.
     LEA_leaks_order(<<b1,ordID>,b2,<g,pkLEA>,skJ>) @ #i
  ==>
      ( Ex a sig #j #k #l.
          J_sen_GRS2_to_LEA(a,<b1,ordID>,b2,<g,pkLEA>,sig) @ #j
      &   Register_sk_J(skJ) @ #k
      &   Register_pk_LEA(pkLEA) @ #l
      &   j<i
      &   k<i
      &   l<i
      )
"

/*************************************************************
*   2.2 CID delivery
*************************************************************/
/*************************************************************
*   2.2.1 Functionality
*************************************************************/

lemma cid_delivery_is_functional:
// It is possible that
exists-trace
"
  Ex cid pkV_lt #i #j.
      // APRS sends cid to CIDS
      APRS_sen_CIDD1_to_CIDS(cid,pkV_lt) @ #i
      // and the vehicle finds it.
  &   Vehicle_finds_cid_and_stores_it(cid,pkV_lt) @ #j
  &   i<j
"
```

```
471
472 /*****************************************************
473 *    2.2.2 Security
474 *****************************************************/
475
476 lemma cid_is_not_known_by_the_adversary:
477 all-traces
478 "
479   // Whenever APRS sends cid to CIDS
480   All cid pkV_lt #i.
481       APRS_sen_CIDD1_to_CIDS(cid,pkV_lt) @ #i
482   ==>
483       // the adversary will not know the cid.
484       not Ex #i. K(cid) @ #i
485 "
486
487 lemma cid_is_not_known_by_other_vehicles:
488 all-traces
489 "
490   // Whenever APRS sends cid to CIDS
491   All cid pkV_lt #i.
492       APRS_sen_CIDD1_to_CIDS(cid,pkV_lt) @ #i
493   ==>
494       // no other vehicles than the receiver will know the cid.
495       not Ex pkV_lt_other #i.
496           Vehicle_finds_cid_and_stores_it(cid,pkV_lt_other) @ #i
497       &   not (pkV_lt_other=pkV_lt)
498 "
499
500 /*****************************************************
501 *    2.3 Data import
502 *****************************************************/
503 /*****************************************************
504 *    2.3.1 Functionality
505 *****************************************************/
506 /*****************************************************
507 *    2.3.1.1 Data import is functional
508 *****************************************************/
509
510 lemma data_import_is_functional:
511 // It is possible that
512 exists-trace
513 "
514   Ex pseu pkV_lt #i #j.
515       // PIA sends pseudonym and long term identity to APRS
516       PIA_sen_DI_to_APRS(pseu,pkV_lt) @ #i
517       // and APRS receives the both.
518   &   APRS_rec_DI_from_PIA(pseu,pkV_lt) @ #j
519   &   i<j
520 "
521
522 /*****************************************************
523 *    2.3.2 Security
524 *****************************************************/
525 /*****************************************************
526 *    2.3.2.1 Vehicles long term identity is not known by
527 *            the adversary
528 *****************************************************/
529
530 lemma vehicles_long_term_identity_security:
531 all-traces
532 "
533     // Whenever PIA sends pseudonym and long term identity to ARPS
534     All pseu pkV_lt #i.
535         PIA_sen_DI_to_APRS(pseu,pkV_lt) @ #i
536     ==>
537         // it should not be that
538         not ( Ex #j.
539             // the adversary knows the long term identity
540             K(pkV_lt) @ #j
541             // without publishing the long term identity.
542         &   ( not Ex #k.
543                 Adversary_knows_pub_key(pkV_lt) @ #k
544             &   k<j
545             )
546         )
547 "
548
549 /*****************************************************
550 *    2.4 Generic resolution system
551 *****************************************************/
552 /*****************************************************
553 *    2.4.1 Functionality
554 *****************************************************/
```

```
555 /*******************************************************
556 *    2.4.1.1 The system is functional
557 *******************************************************/
558
559 lemma the_system_is_functional:
560 // It is possible that
561 exists-trace
562 "
563   Ex a b1 ordID b2 e1 pub_resID e2 z sig sig_2 sig_3 sig_4 d resID pseu k s cid pkLEA
        ↪ pkV_lt g eta b3 e3 #i #j #k #l #m #n #o #p #q #r #s #t.
564       // LEA sends  resolution permission request to J,
565       LEA_sen_GRS1_to_J(a) @ #i
566       // J sends corresponding resolution order to LEA,
567   &   J_sen_GRS2_to_LEA(a,<b1,ordID>,b2,<g,pkLEA>,sig) @ #j
568       // LEA uses the order to send resolution request to APRS with a pseudonym as an
            ↪ argument,
569   &   LEA_sen_GRS3_to_APRS(<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sig,sig_3) @ #k
570       // APRS receives the request,
571   &   APRS_rec_GRS3_from_LEA(<<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sig,sig_3,'GRS3
        ↪ '>) @ #l
572       // APRD sends PFM1 to PF,
573   &   APRS_sen_GRS4_to_PF(<<b1,ordID>,<e1,pub_resID>,<b2,e2,k,pkV_lt,z,k,cid,s>>,sig_
        ↪ 2) @ #l
574       // PF receives the PFM1,
575   &   PF_rec_GRS4_from_APRS(<b1,ordID>,<e1,pub_resID>) @ #m
576       // APRS sends resolution result to LEA,
577   &   APRS_sen_GRS6_to_LEA(<d,resID,pseu>,<eta,pkV_lt>) @ #n
578       // LEA receives the resolution result,
579   &   LEA_rec_GRS6_from_APRS(<d,resID,pseu>,<eta,pkV_lt>) @ #o
580       // J receives information of the resolution,
581   &   J_receives_information_of_res(ordID,resID) @ #p
582       // J sends order to publish PFM2,
583   &   J_sen_GRS7_to_APRS(b3,ordID,resID) @ #q
584       // APRS receives the order,
585   &   APRS_rec_GRS7_from_J(<b3,ordID>,resID) @ #r
586       // APRD sends PFM2 to PF,
587   &   APRS_sen_GRS8_to_PF(<b3,ordID>,e3,b2,<e2,pub_resID>,pkV_lt,z,k,cid,s,sig_4) @ #r
588       // PF receives the PFM2,
589   &   PF_rec_GRS8_from_APRS(<b3,ordID>,pub_resID) @ #s
590       // and the target vehicle finds the target data inside the PFM2
591   &   Vehicle_finds_target_data(pkV_lt,z) @ #t
592       // in the correct order.
593   &   i<j
594   &   j<k
595   &   k<l
596   &   l<m
597   &   m<n
598   &   n<o
599   &   n<p
600   &   p<q
601   &   q<r
602   &   r<s
603   &   s<t
604 "
605
606 /*******************************************************
607 *    2.4.1.2 If PFM1 and PFM2 with the same pub
608 *            resolution ID are published,
609 *            then PFM1 is before PFM2
610 *******************************************************/
611
612 lemma if_PFM1_and_PFM2_with_the_same_resID_are_published_then_PFM1_is_before_PFM2:
613 all-traces
614 "
615   // Whenever PFM1 and PFM1 with the same resolution ID are published,
616   All b1 ordID e1 pub_resID b2 b3 e2 e3 z sig sig_2 k s cid pkV_lt #i #j.
617       APRS_sen_GRS4_to_PF(<<b1,ordID>,<e1,pub_resID>,<b2,e2,k,pkV_lt,z,k,cid,s>>,sig_
            ↪ 2) @ #i
618   &   APRS_sen_GRS8_to_PF(<b3,ordID>,e3,b2,<e2,pub_resID>,pkV_lt,z,k,cid,s,sig) @ #j
619   ==> // then PFM1 is before PFM2.
620       i<j
621 "
622
623 /*******************************************************
624 *    2.4.1.3 Js order to send PFM2 messages leads to
625 *            publishing a correct PFM2
626 *******************************************************/
627
628 lemma Js_order_to_send_PFM2_messages_leads_to_publishing_a_correct_PFM2:
629
630 // It is possible that
631 exists-trace
632 "
633   // when J orders to send PFM2 messages for ordID
```

```
634     All ordID b3 resID #i.
635         J_sen_GRS7_to_APRS(b3,ordID,resID) @ #i
636     ==>
637         Ex b1 e1 pub_resID #j #k.
638             // there was PFM1 for the ordID
639             PF_rec_GRS4_from_APRS(<b1,ordID>,<e1,pub_resID>) @ #j
640             // and PFM2 for the ordID is published.
641         &   PF_rec_GRS8_from_APRS(<b3,ordID>,pub_resID) @ #k
642         &   j<i
643         &   i<k
644 "
645
646 /*************************************************************
647 *    2.4.1.4 If a PFM2 is published, then J
648 *            sent order to send PFM2 messages before
649 *            publishment
650 *************************************************************/
651
652 lemma If_a_PFM2_is_published_then_J_sent_order_to_send_PFM2_messages_before_
        ↪ publishment:
653 all-traces
654 "
655     // Whenever PF receives PFM2
656     All b3 ordID pub_resID #i.
657         PF_rec_GRS8_from_APRS(<b3,ordID>,pub_resID) @ #i
658     ==>
659         Ex resID #j.
660             // J sent order to send PFM2 messages
661             J_sen_GRS7_to_APRS(b3,ordID,resID) @ #j
662             // before the PFM2.
663         &   j<i
664
665 "
666
667 /*************************************************************
668 *    2.4.1.5 LEA with a resolution order can use the
669 *            order to send multiple different
670 *            resolution requests
671 *************************************************************/
672
673 lemma LEA_can_use_an_order_to_resolve_many_times:
674 // It is possible that
675 exists-trace
676 "
677     Ex a b1 ordID b2 d d_2 resID resID_2 pseu pseu_2 g pkLEA sig sig_2 #i #j #k.
678         // J sends resolution order to LEA,
679         J_sen_GRS2_to_LEA(a,<b1,ordID>,b2,<g,pkLEA>,sig) @ #i
680         // LEA sends a resolution request 1,
681     &   LEA_sen_GRS3_to_APRS(<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sig,sig_2) @ #j
682         // LEA sends a resolution request 2,
683     &   LEA_sen_GRS3_to_APRS(<b1,ordID>,b2,<g,pkLEA>,<d_2,resID_2,pseu_2>,sig,sig_2) @ #
            ↪ k
684         // resolution request 1 is sent before resolution request 2,
685     &   j<k
686         // J sent the resolution order before resolution request 1,
687     &   i<j
688         // and resolution request 1 is not the same than resolution request 2.
689     &   not (d = d_2)
690     &   not (resID = resID_2)
691 "
692
693 /*************************************************************
694 *    2.4.1.6 Adversary knows b1 and e1 after
695 *            a PFM1
696 *************************************************************/
697
698 lemma adversary_knows_b1_and_e1_after_a_PFM1:
699 // It is possible that
700 exists-trace
701 "
702     // when the APRS sends a PFM1
703     All b1 ordID e1 pub_resID b2 e2 k pkV_lt z cid s sig_2 #i.
704         APRS_sen_GRS4_to_PF(<<b1,ordID>,<e1,pub_resID>,<b2,e2,k,pkV_lt,z,k,cid,s>>,sig_
                ↪ 2) @ #i
705     ==>
706         Ex #j.
707             // the adversary will know b1 and e1.
708             K(b1) @ #j
709         &   K(e1) @ #j
710         &   i<j
711 "
712
713 /*************************************************************
714 *    2.4.1.7 Adversary knows b2, b3, e2 and
```

```
*           e3 after a PFM2
**********************************************************/

lemma adversary_knows_b2_b3_e2_and_e3_after_a_PFM2:
// It is possible that
exists-trace
"
  // when the APRS sends a PFM2
  All b3 ordID e3 b2 e2 pub_resID pkV_lt z k cid s sig #i.
      APRS_sen_GRS8_to_PF(<b3,ordID>,e3,b2,<e2,pub_resID>,pkV_lt,z,k,cid,s,sig) @ #i
  ==>
      Ex #j.
          // the adversary will know b2, b3, e2, and e3.
          K(b2) @ #j
      &   K(b3) @ #j
      &   K(e2) @ #j
      &   K(e3) @ #j
      &   i<j
"

/*******************************************************
*   2.4.1.8 A vehicle can read target data published
*           for it
*******************************************************/

lemma a_vehicle_can_read_target_data_published_for_it:
// It is possible that
exists-trace
" // when the APRS publishes a target data
  All b3 ordID e3 b2 e2 pub_resID pkV_lt z k cid s sig #i.
      APRS_sen_GRS8_to_PF(<b3,ordID>,e3,b2,<e2,pub_resID>,pkV_lt,z,k,cid,s,sig) @ #i
  ==> // the target can find it.
      Ex #j.
          Vehicle_finds_target_data(pkV_lt,z) @ #j
"

/*******************************************************
*   2.4.2 Security
*******************************************************/
/*******************************************************
*   2.4.2.1 Pseudonym resolution leads to publishing
*           a PFM1
*******************************************************/

lemma res_leads_to_publishing_a_PFM1:
all-traces
"
  // Whenever the LEA receives a resolution result,
  All d resID pseu eta pkV_lt #i.
      LEA_rec_GRS6_from_APRS(<d,resID,pseu>,<eta,pkV_lt>) @ #i
  &   not ( Ex ad_data #j. Malicious_LEA_sends_res_request_to_APRS_with_data_from_
      ↪ adversary(ad_data) @ #j )
  ==>
      // the corresponding PFM1 was published.
      Ex b1 d resID ordID b2 g sig e1 e2 z sig_2 sig_3 pub_resID pseu k s cid pkLEA
          ↪ pkV_lt  #j #k #l.
          LEA_sen_GRS3_to_APRS(<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sig,sig_2) @ #j
      &   APRS_rec_GRS3_from_LEA(<<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sig,sig_2,'
          ↪ GRS3'>) @ #k
      &   APRS_sen_GRS4_to_PF(<<b1,ordID>,<e1,pub_resID>,<b2,e2,k,pkV_lt,z,k,cid,s>>,
          ↪ sig_3) @ #k
      &   PF_rec_GRS4_from_APRS(<b1,ordID>,<e1,pub_resID>) @ #l
      &   j<k
      &   k<l
      &   l<i
"

/*******************************************************
*   2.4.2.2 If a PFM1 is published, then
*           resolution happened
*******************************************************/

lemma if_a_PFM1_is_published_then_res_happened:
all-traces
"
  // Whenever a PFM1 is published,
  All b1 ordID e1 pub_resID #i.
      PF_rec_GRS4_from_APRS(<b1,ordID>,<e1,pub_resID>) @ #i
  &   not ( Ex ad_data #j. Malicious_LEA_sends_res_request_to_APRS_with_data_from_
      ↪ adversary(ad_data) @ #j )
  ==>
      // the corresponding resolution happened before that.
      Ex g b2 e2 z sig sig_2 sig_3 d resID pseu k s cid pkLEA pkV_lt ordID pub_resID #
          ↪ j #k.
```

```
793             LEA_sen_GRS3_to_APRS(<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sig,sig_2) @ #j
794         &   APRS_rec_GRS3_from_LEA(<<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sig,sig_2,'
                ↪ GRS3'>) @ #k
795         &   APRS_sen_GRS4_to_PF(<<b1,ordID>,<e1,pub_resID>,<b2,e2,k,pkV_lt,z,k,cid,s>>,
                ↪ sig_3) @ #k
796         &   j<k
797         &   k<i
798 "
799
800 /*******************************************************
801 *   2.4.2.3 Only LEA with permission from J can
802 *           resolve
803 *
804 *           When J signs an order, LEA ID is also signed.
805 *******************************************************/
806
807 lemma stolen_orders_cannot_be_used_for_res:
808 all-traces
809 "
810   // Whenever a malicious LEA sends an resolution request with a stolen order
811   All ad_data #i.
812       Malicious_LEA_sends_res_request_to_APRS_with_data_from_adversary(ad_data) @ #i
813   &   not ( Ex request #i. LEA_leaks_request(request) @ #i )
814
815   ==>
816       // the APRS does not accept the request.
817       not ( Ex data #j.
818           APRS_rec_GRS3_from_LEA(data) @ #j
819       &   ad_data=data
820       )
821 "
822
823 /*******************************************************
824 *   2.4.2.4 a is not known by the adversary
825 *******************************************************/
826
827 lemma a_is_not_known_by_the_adversary:
828 all-traces
829 "
830   // Whenever a LEA sends a resolution permission request alpha to a J
831   All a #i.
832       LEA_sen_GRS1_to_J(a) @ #i
833   ==>
834       // it should not be that
835       not ( Ex #j.
836           // the adversary knows the alpha.
837           K(a) @ #j
838       )
839 "
840
841 /*******************************************************
842 *   2.4.2.5 b1 is not known by the adversary before
843 *           a PFM1 (LEA1 can use the b1 for many
844 *           queries)
845 *******************************************************/
846
847 lemma b1_is_not_known_by_the_adversary_before_a_PFM_1:
848 all-traces
849 "
850   // Whenever resolution happens
851   All d resID pseu a b1 ordID b2 g pkLEA sig eta pkV_lt #i #j.
852   (
853   (
854       LEA_rec_GRS2_from_J(a,<b1,ordID>,b2,<g,pkLEA>,sig) @ #i
855   &   LEA_rec_GRS6_from_APRS(<d,resID,pseu>,<eta,pkV_lt>) @ #j
856   &   not ( Ex order #i. LEA_leaks_order(order) @ #i )
857   &   not ( Ex request #i. LEA_leaks_request(request) @ #i )
858   )
859   ==> // it should not be that
860       (
861       not ( Ex #m.
862           // the adversary knows b1
863           K(b1) @ #m
864           // without a PFM1 from APRS containing b1.
865       &   not (Ex e1 pub_resID e2 z sig_2 k s cid pkV_lt #n. APRS_sen_GRS4_to_PF(<<b1,
                ↪ ordID>,<e1,pub_resID>,<b2,e2,k,pkV_lt,z,k,cid,s>>,sig_2) @ #n)
866       )
867       )
868   )
869 "
870
871 /*******************************************************
872 *   2.4.2.6 b2 is not known by the adversary before
873 *           a PFM2
```

```
874 ******************************************************/
875
876 lemma b2_is_not_known_by_the_adversary_before_PFM_2:
877 all-traces
878 "
879   // Whenever resolution happens
880   All d resID pseu b1 ordID b2 g pkLEA sig sig_2 eta pkV_lt #i #j.
881       LEA_sen_GRS3_to_APRS(<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sig,sig_2) @ #i
882   &   LEA_rec_GRS6_from_APRS(<d,resID,pseu>,<eta,pkV_lt>) @ #j
883   &   not ( Ex order #i. LEA_leaks_order(order) @ #i )
884   &   not ( Ex request #i. LEA_leaks_request(request) @ #i )
885   ==>
886       // it should not be that
887       not ( Ex #m.
888           // the adversary knows a b2
889           K(b2) @ #m
890           // without a PFM 2 from the APRS containing the b2.
891       &   not (Ex b3 e3 e2 pub_resID pkV_lt k z cid s sig_3 #n. APRS_sen_GRS8_to_PF(<
                ↪ b3,ordID>,e3,b2,<e2,pub_resID>,pkV_lt,z,k,cid,s,sig_3) @ #n)
892       )
893 "
894
895 /******************************************************
896 *   2.4.2.7 b3 is not known by the adversary before
897 *           a PFM2
898 ******************************************************/
899
900 lemma b3_is_not_known_by_the_adversary_before_PFM_2:
901 all-traces
902 "
903   // Whenever resolution happens and J decides to inform
904   All d resID pseu b1 ordID b2 g pkLEA sig sig_2 eta pkV_lt b3 #i #j #k.
905       LEA_sen_GRS3_to_APRS(<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sig,sig_2) @ #i
906   &   LEA_rec_GRS6_from_APRS(<d,resID,pseu>,<eta,pkV_lt>) @ #j
907   &   J_sen_GRS7_to_APRS(b3,ordID,resID) @ #k
908   ==>
909       // it should not be that
910       not ( Ex #m.
911           // the adversary knows b3
912           K(b3) @ #m
913           // without a PFM 2 from the APRS containing b3.
914       &   not (Ex e3 e2 pub_resID pkV_lt k z cid s sig_3 #n. APRS_sen_GRS8_to_PF(<b3,
                ↪ ordID>,e3,b2,<e2,pub_resID>,pkV_lt,z,k,cid,s,sig_3) @ #n)
915       )
916 "
917
918 /******************************************************
919 *   2.4.2.8 g is not known by the adversary
920 ******************************************************/
921
922 lemma g_is_not_known_by_the_adversary:
923 all-traces
924 " // Whenever a J sends a resolution order gamma to LEA
925   All a b1 ordID b2 g pkLEA sig #i.
926       J_sen_GRS2_to_LEA(a,<b1,ordID>,b2,<g,pkLEA>,sig) @ #i
927   &   not ( Ex order #i. LEA_leaks_order(order) @ #i )
928   &   not ( Ex request #i. LEA_leaks_request(request) @ #i )
929
930   ==>
931       // it should not be that
932       not ( Ex #j.
933           // the adversary knows the g.
934           K(g) @ #j
935       )
936 "
937
938 /******************************************************
939 *   2.4.2.9 d is not known by the adversary
940 ******************************************************/
941
942 lemma d_is_not_known_by_the_adversary:
943 all-traces
944 " // Whenever a LEA sends a d to the APRS
945   All b1 ordID b2 g pkLEA d resID pseu sig sig_2 #i.
946       LEA_sen_GRS3_to_APRS(<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sig,sig_2) @ #i
947   &   not ( Ex order #i. LEA_leaks_order(order) @ #i )
948   &   not ( Ex request #i. LEA_leaks_request(request) @ #i )
949   ==>
950       // it should not be that
951       not ( Ex #j.
952           // the adversary knows the d.
953           K(d) @ #j
954       )
955 "
```

```
956
957  /*****************************************************
958  *   Lemma "e1 is not known by the adversary before
959  *   a PFM 1" is not relevant since the e1 is generated
960  *   by the APRS just before sending it to the PF
961  *   (in the rule APRS_2 the e1 is fresh).
962  *****************************************************/
963
964  /*****************************************************
965  *   2.4.2.10 e2 is not known by the adversary before
966  *            a PFM 2
967  *****************************************************/
968
969  lemma e2_is_not_known_by_the_adversary_before_PFM_2:
970  all-traces
971  " // Whenever the APRS generates an e2
972    All b1 ordID e1 pub_resID b2 e2 pkV_lt z k cid s sig_2 #i.
973        APRS_sen_GRS4_to_PF(<<b1,ordID>,<e1,pub_resID>,<b2,e2,k,pkV_lt,z,k,cid,s>>,sig_
                ↪ 2) @ #i
974    ==>
975        // it should not be that
976        not ( Ex #m.
977            // the adversary knows the e2
978            K(e2) @ #m
979            // without a PFM 2 from the APRS containing the e2.
980        &   not (Ex e3 b3 ordID_2 pub_resID_2 pkV_lt_2 k_2 z_2 cid_2 s_2 sig_3 #n. APRS_
                ↪ sen_GRS8_to_PF(<b3,ordID_2>,e3,b2,<e2,pub_resID_2>,pkV_lt_2,z_2,k_2,cid_
                ↪ 2,s_2,sig_3) @ #n)
981        )
982  "
983
984  /*****************************************************
985  *   Lemma "e3 is not known by the adversary before
986  *   a PFM 2" is not relevant since the e3 is generated
987  *   by the APRS just before sending it to the PF
988  *   (in the rule APRS_4 the e3 is fresh).
989  *****************************************************/
990
991  /*****************************************************
992  *   2.4.2.11 Target data z is not known by the adversary
993  *****************************************************/
994
995  lemma z_is_not_known_by_the_adversary:
996  all-traces
997  " // Whenever the APRS publishes a PFM 2
998    All b1 ordID e1 pub_resID b2 e2 pkV_lt z k cid s sig_2 #i.
999        APRS_sen_GRS4_to_PF(<<b1,ordID>,<e1,pub_resID>,<b2,e2,k,pkV_lt,z,k,cid,s>>,sig_
                ↪ 2) @ #i
1000   ==>
1001       // it should not be that the adversary knows the target data z.
1002       not Ex #i. K(z) @ #i
1003 "
1004
1005 /*****************************************************
1006 *   2.4.2.12 eta is not known by the adversary
1007 *****************************************************/
1008
1009 lemma eta_is_not_known_by_the_adversary:
1010 all-traces
1011 " // Whenever the APRS generates a resolution result eta
1012   All d resID pseu eta pkV_lt #i.
1013       APRS_sen_GRS6_to_LEA(<d,resID,pseu>,<eta,pkV_lt>) @ #i
1014   ==>
1015       // it should not be that the adversary knows the result eta
1016       not Ex #i. K(eta) @ #i
1017       &  not Ex #j. K(pkV_lt) @ #j
1018 "
1019
1020 /*****************************************************
1021 *   2.4.2.13 An resolution request is handled only
1022 *            once (avoiding replay attacks)
1023 *****************************************************/
1024
1025 lemma an_res_request_is_handled_only_once:
1026 all-traces
1027 "
1028   // Whenever a LEA sends a resolution request to the APRS,
1029   All b1 ordID b2 g pkLEA d resID pseu sig sig_2 #i.
1030       LEA_sen_GRS3_to_APRS(<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sig,sig_2) @ #i
1031   ==>
1032       // the APRS handles the request only once.
1033       not Ex pkLEA_2 sig_1 sig_3 #j #k.
1034           APRS_rec_GRS3_from_LEA(<<b1,ordID>,b2,<g,pkLEA_2>,<d,resID,pseu>,sig_1,sig_
                ↪ 3,'GRS3'>) @ #j
```

```
1035        &    APRS_rec_GRS3_from_LEA(<<b1,ordID>,b2,<g,pkLEA_2>,<d,resID,pseu>,sig_1,sig_
                 ↪ 3,'GRS3'>) @ #k
1036        &    not ( #j = #k )
1037
1038    "
1039
1040    /*****************************************************
1041    *    2.4.2.14 If a resolution happens with a pseudonym
1042    *             as a resolution argument the corresponding
1043    *             result contains always the correct holder
1044    *             long term identity
1045    *****************************************************/
1046
1047    lemma resolution_result_corresponds_to_resolution_argument:
1048    all-traces
1049    "
1050      // Whenever
1051      All pseu pkV_lt b1 ordID b2 g pkLEA d resID sig sig_3 eta pkV_lt_2 #i #j #k.
1052           // PIA sends pseudonym and the corresponding long term identity to the APRS,
1053           PIA_sen_DI_to_APRS(pseu,pkV_lt) @ #i
1054           // LEA sends resolution reguest with the pseudonym,
1055      &    LEA_sen_GRS3_to_APRS(<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sig,sig_3) @ #j
1056           // and LEA receives a resolution result
1057      &    LEA_rec_GRS6_from_APRS(<d,resID,pseu>,<eta,pkV_lt_2>) @ #k
1058      ==>
1059           // the long-term identity in the result corresponds to the argument pseudonym.
1060           pkV_lt=pkV_lt_2
1061    "
1062
1063    /*****************************************************
1064    *    2.4.2.15 Only authorized LEA can resolve
1065    *****************************************************/
1066
1067    lemma only_authorized_LEA_can_resolve:
1068    all-traces
1069    "
1070      // Whenever
1071      All d resID pseu eta pkV_lt #i.
1072           // a LEA receives a resolution result
1073           LEA_rec_GRS6_from_APRS(<d,resID,pseu>,<eta,pkV_lt>) @ #i
1074           // and no resolution requests have been leaked,
1075      &    not ( Ex request #j. LEA_leaks_request(request) @ #j )
1076      ==>
1077           Ex a b1 ordID b2 g pkLEA sig sig_3 #j #k.
1078                // the judge authorized the resolution
1079                J_sen_GRS2_to_LEA(a,<b1,ordID>,b2,<g,pkLEA>,sig) @ #j
1080                // and the LEA used the authorization to resolve.
1081           &    LEA_sen_GRS3_to_APRS(<b1,ordID>,b2,<g,pkLEA>,<d,resID,pseu>,sig,sig_3) @ #k
1082           &    #j<#i
1083    "
1084
1085    end
```