Proceedings of the 18th Mediterranean Electrotechnical Conference
MELECON 2016, Limassol, Cyprus, 18-20 April 2016

Secure model to share data in Intelligent Transportation Systems

Cătălin Goşman*, Tudor Cornea*, Ciprian Dobre*, Constandinos X. Mavromoustakis†, George Mastorakis‡

*Department of Computer Science,
University Politehnica of Bucharest
Spl. Independentei 313, Bucharest, Romania
{catalin.gosman, tudor.cornea}@cti.pub.ro,
ciprian.dobre@cs.pub.ro

†Department of Computer Science,
University of Nicosia,
46 Makedonitissas Avenue, 1700
Nicosia, Cyprus
mavromoustakis.c@unic.ac.cy

‡Technological Educational
Institute of Crete
Estavromenos 71500, Heraklion,
Crete, Greece
gmastorakis@staff.teicrete.gr

***Abstract*—Cars have become essential elements of modern life. But nowadays the increasing number of cars also leads to problems: pollution, traffic jams leading to wasted time spent in traffic. Intelligent Transportation Systems (ITS) demonstrate innovative services relating to different modes of transport and traffic management, and enable various users to be better informed and make safer, more coordinated, and smart use of transport networks. To capture accurate models of the traffic, many ITS rely on users to willingly contribute data captured by sensors made available by personal mobile devices. This way, mobile devices become ubiquituous sensing devices, able to cover any area where users are present. However, for the user, sharing such data can quickly lead to privacy and security issues, such as disclosing location to unintended audience. Obviously, all ITS application should include security as one priority requirement, but unfortunately not many do so. In this paper, we answer the question: "How could ITS applications, even existing ones, include privacy guarantees they can show to the users contributing their private data?". We propose a model where users can define data sharing policies associated with their personal information flows captured by an ITS application. Under this model, ITS applications might need to use specific data under various quality and context-based constraints, while the user-oriented security policies could specify additional usage constraints, and we propose a solution to mediate between these two sides. We evaluate a pilot security implementation of the model, and evaluate it under real-world assumptions.**

I. Introduction

Intelligent transportation systems (ITS) rely on a level of communication that facilitates data exchange between vehicles and between vehicles and the road infrastructure (or data centers in which traffic information is aggregated in order to obtain applications that can be used in order to optimize / control the traffic).

Most current Intelligent Transport Systems today are being developed as proprietary, closed-source solutions. Besides the fact that this leads to problems related to a major lack of interconnectivity between such systems, this also means users have no guarantees about the security mechanisms being used, and no control over the data sharing process. For example, a user helping a navigation application (e.g., Waze, Google Traffic) has little knowledge who and under what terms could potentially have access to his data. At a first glance, this would not be an important problem. However, sharing location data could potentially lead to the user disclosing sensitive information about driving routes or driving preferences.

978-1-5090-0058-6/16/$31.00

The idea of sharing data in ITS has branched into two data collection models [1]:

1) In the participatory model the user is actively/willingly sending data, and has full control over when and how his data is shared.
2) In the opportunistic model the user is sensing data without being actively involved in the process.

What is common in both models is the fact that users might store data that contains sensitive information for them, like location, places visited, stops during the day etc. Therefore, the process of sharing information, sometimes sensitive, may raise some questions regarding users' privacy needs and requirements: Who owns the data, and under which conditions this data is shared with other entities in ITS, who benefits from the data collection process? In many of the current approaches, users have little control over the data collection process, cannot influence the information sharing process. *Users willingly lose control and access to their personal shared data.*

The paper proposes a new model for users to define data sharing policies for their personal information that is captured by an ITS. We assume the collection of context data (timestamp, location, possible speed, hashed identity of collecting device, etc.) that are used as parameters of the security policy, and are combined using logical operators. We evaluate formally the security policy model, and even evaluate its feasibility under a pilot implementation. We developed an ITS navigation system, where a security manager imposes the security policies defined by users in their mobile clients.

The rest of the paper is structured as follows: In Section II we analyse security in various ITS applications, and show how they compare, if any, to our proposal. Next, in Section III we present our approach, followed by a formal validation in Section V. We present our pilot implementation and experimental results in Section VI. Section VII presents conclusions and future work.

II. Related work

One of the biggest challenges even today for ITS remains security. We stopped below at some of the most important approaches in that direction.

In the PRISM project [2], participating nodes (mobile devices) are registering to a server that keeps track of the participants, and offers back to users information such as location of other participants. In PRISM security mainly ensures that users not registered cannot access the gathered data. The drawback of PRISM is that it does not ensure anonymity, neither for participants that ask data about others, neither for participants that share data.

A solution that offers a certain degree of privacy is AnonySense [3]. The list of aspects, as defined by authors, regarding contextual information that can be requested, is published on the server. Participating nodes can download the list and filter from it certain aspects they want to share with others. This approach has the advantage that users reveal nothing related to the information they want to share with the server, as they are accepting or declining a chunk of aspects from the list published on the server from which participants can ask contextual information. The disadvantage is that user identity is not protected once he decides to share one aspect with the public list published on the server.

PEPSI [4] developed a system with anonymity requirements, for ITS applications needing to share data and discover relevant contextual information out of it. The system is centralized, and in order to keep users' anonymity, it uses a Registration Authority. This entity collects contextual demands from users, and offers in exchange cryptographic material. Therefore, participants that want to obtain contextual information can make queries to the PEPSI platform in an encrypted manner. The disadvantage of the solution is that the problem has now been moved with the Registration Authority, that knows in advance all demands regarding contextual information, and also the identities of all those users that want to access contextual information. Therefore, authors assume in PEPSI that the Registration Authority is always reliable component – which is not always true.

In our case, the starting point for us was represented by the MobiWay project. The nationally-funded project aims to construct a platform for connecting and sharing data between ITS actors (systems, applications, users), supporting information management on a large scale. A special component in the project is dedicated to connecting various ITS sources together, facilitating the introduction and maintenance of ITS services, achieved through mediation, service specific selection and composition. After data is gathered, we store it in Data Vaults (DV), which are logically distinct data stores (i.e., per user or group of users) that declare and impose a set of policies or rules for each individual's data [5]. Before the data can be extracted and processed in MobiWay, a Security Engine is first queried, and a decision is made about the availability of data with regard to that particular processing step. The Security Engine enforces the requirements of the DVs. To achieve this, every processing request made by an ITS service triggers a policy check. Policies are further extracted on request from a Policy Store. The Security Engine is also responsible for preserving the privacy of all participants. For this, we introduce in this paper our solution for preserving privacy.

III. Security Policies in ITS

In the following Sections, we present our proposed security and privacy model. This model is designed for ITS applications which assume that various users cooperate and work together to have a sense of traffic conditions, for example. This is the case with applications such as car sharing, carpooling, taxi management applications, or simple navigators like Waze and Google Traffic.

We assume the existence of several sources (S_i, for i in the set of all sources) sensing and sending traffic data to a collecting platform, where it is potentially aggregated into traffic models, or used directly by various ITS applications and services (A_i, for i in the set of all applications needing the collected data). In this case, a security policy has to specify permissions oriented towards the data source (that defines the access control for the data). As specification, the security policy has to specify (1) sensitivity of the data and (2) under what conditions they can be accessed. Condition (1) means that data is classified according to the sensing context. Condition (2) means that data can be accessed differently by various actors, or under different usage patterns.

We assume that such security policies are applied at the level of each Data Vault (DV_i, for i in the set of all Data Vaults defined for all sources of data S_i) – we recall that the Data Vault is the logical data storage structure used to separate the data being collected from each particular source. For DV_i, source S_i has full access at the stored information, and can control and specify access restrictions in the form of security policies.

The problem with previous security solutions for ITS is that privacy considerations are usually specified at the level of a group of sources (e.g., all sources belonging to a group accept same similar privacy requirements, usually based on geographical collocation). For example, if an application cannot see information about traffic in a particular area, no data sensed by any car in that area will be disclosed, while it potentially see other data from cars in other areas.

In our case, we want even a user having installed an ITS application on his mobile device (source S_i) to be able to change preferences regarding sharing policy with an ITS collecting platform (fine granularity). In the same time, we want to be able to specify security policies even at group level. For example, a taxi company could decide that all data belonging to its fleet of cars is to be shared under one unique sharing policy. A compromise between the two approaches must be found.

A. Privacy concerns – the perspective of the data collector

Most often, a source S_i can collect sensitive information from which a third partially could benefit. Different data fields

being collected could disclose sensitive information: location could disclose route patterns or potential traffic events [6]; GSM and/or Wireless data could disclose crowd behavior [7].

A problem perceived by users contributing with data in ITS is the lack of access control options for their stored information. A user cannot influence how and by what application / service A_i the personally collected data will be used. In principle, A_i should guarantee confidentiality and data protection for all information stored by any source S_i.

B. Privacy concerns – the perspective of the application

The ITS application or service should consider all privacy requirements of the S_i. Otherwise, sources can become reluctant in contributing with information – for example, if a driver can get fired because the system lets his employee know about his whereabouts, than, if he has the alternative to notcontribute with his data, he will do so. We consider here the case where incentives are not motivating enough the driver (like in the case that a company would force all drivers use the application).

Thus, if privacy requirements are not treated properly, ITS providers may be confronted with having only small volumes of data, but also with legal constraints regarding user data privacy violations.

Having in mind the privacy demands that manifest both on sources S_i, but also on applications / services A_i, we can conclude that developing an ITS service that satisfies each individual user privacy constraint is a complex problem, compared for example with a service using a fix set of privacy constraints applied only at the group level for collecting sources.

In order to focus on each individual privacy demands, a way to control access to their data is imperative. Our proposed solution uses *security policies*. Security policies are applied at the DV_i level. The DV_i acts as a proxy for a source S_i, allowing it to control access to its data in ITS. In this manner, the DV_i has multiple functions: it ensures data protection and access control to stored information, it maintains flexibility regarding the security policies accepted, forcing data to be shared only according to the security policies accepted at the DV_i level.

Usually, a security policy determines a binary behavior for an application A_i; either accepts the policy considering all its constraints, or do not use it at all. Also, A_i can work and accept data coming from several sources S_i. The data can be even redundant (e.g., take location data coming from both participants in traffic, and sensors located at the road level, even if both these sources deliver similar information). Of course, this leads to potential problem related to consistency and coherence. And, this leads to potential conflicts for the privacy policies for these different sources, that DV_i has to solve.

IV. Implementing security policies in ITS eco-systems

We now define how a security policy should look like for an ITS. First, a security policy ensures that data access to DV_i is secured and controlled only by the source S_i (the 'originator' of information). The mechanism defined for specifying what and who can access (the 'policy') should be flexible enough – for various situations (defined as the context flowing in time [8]), context could be 'secured' differently (e.g., location could be hidden for particular areas in town, or should be shared differently considering various time moments). The policy should be easily adapted to such changes in how S_i wants the data to be shared. Such a change should be initiated by the user, or should automatically be made possible by the system (based on various triggers). For the first case, we consider the user to not be necessarily an expert in the field – any driver should easily understand the logic of sharing, and should easily define or modify his current privacy sharing policy.

All in all, the aspects used in defining and managing a security policy should affect both the source S_i that shares data (according to the security policy), and also the ITS application / service A_i in need of that data. The developer of an ITS application or service A_i could be well aware of the security policies in place, how they are defined, how they affect the sharing of the data. However, this is not fully needed. For example, we assume that older ITS applications should still work, even after introducing a new security policy. To solve this, we further assume an intermediary component, namely the *ITS collecting and sharing platform*. This component organize the Data Vaults, and ensures the proper enforcement for all security policies that are valid at any given point in time.

With this, the ITS system becomes a combination of three actors (see Figure 1): We have several sources S_i that can collect data on-the-ground. This data and related security sharing policies are further aggregated by the collecting platform, where information belonging to each source is logically separated in Data Vaults (DV_i). Finally, on top, we have applications and services A_i using the data (either individual data, or data aggregated from several sources). If A_i needs data, it can actively request information about the security policies (for example, to decide what sources are to be trusted).

With this, the platform will allow the sharing of data stored in DV_i only if the request coming from A_i is in accordance with the privacy requirements defined by source S_i.

However, there are situations when it is necessary to consider a deviation from the strict rule of applying the security policy. For example, a medical emergency service could gain unrestricted access to data if this would lead to lives being saved, no matter what the user sharing the data implies. In such exceptional situations, security policies could be neglected so that data in DV_i is shared. To define such cases, we argue that control flexibility should be influenced by defined events. An inner event could be a safety one, potentially generated by the source (e.g., please inform all drivers if a pothole is detected). External events can be generated by sources S_j external to the source S_i, or by an application or service A_i. In order to ensure a flexible control, source S_i can define what third party entities (other sources or applications) can access its data unrestricted. To develop this, we assume a Role Based Accessed Control

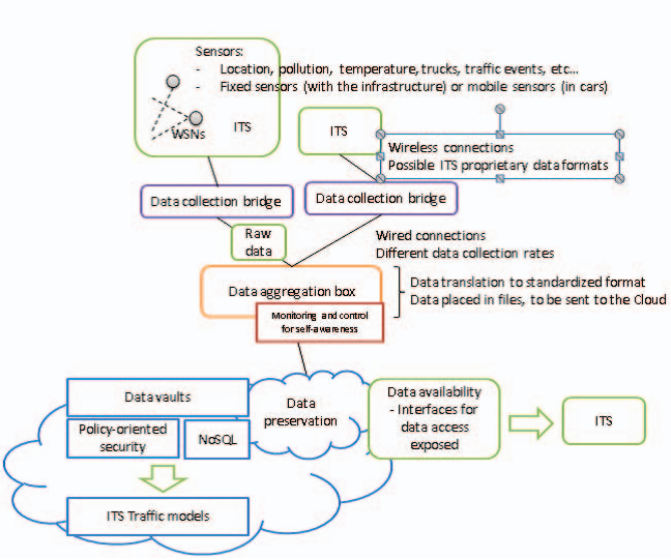

Fig. 1. An overview of the ITS eco-system, as defined in MobiWay.

(RBAC) model can be used.

At the source level, a security policy can be defined using annotations on fields that are part of the information shared by source S_i. Using annotations, a user can specify what data is sensitive according to the accepted security policy. Below is a generic example for using annotation at field level. Fields C_i, C_{i+1}, C_j, C_n are defined as being sensitive by the source S_i. In accordance with policy P_i accepted at DV_i level, these fields will be shared according to a predicate defined by annotation S (see Figure 2). For example, annotation here could dictate that fields are either shared or not (boolean), or could state that two particular fields will never be shared simultaneously with the same application. For the second case, we consider situations where, for example, data coming from microphone coupled with location information could lead to situations where applications could infer critical events (confidential discussions). Or, the annotation could state conditions regarding the applications – the data could be shared differently, depending on the application requesting it.

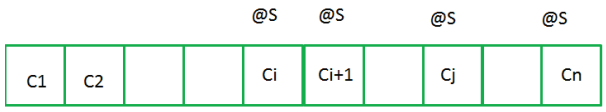

Fig. 2. Representation of a basic security policy applied at the field level.

Security policies are defined at the platform level P, a central entity part of ITS. When a source S_i accepts a security policy defined in platform P, the source should understand the technical implications even in case of an unexperimented user, with little understanding regarding security policies interactions. Accordingly, we define at the platform level a set of default simple security policies that users can understand: share or not location, share or not WiFi data, etc. This simple textual representation of policies can be aggregated into complex sharing filters on the platform. For advanced users, the security policy can be defined on a fine-grained level: for these fields, apply these particular sharing operators (predicates). Finally, the user can apply several such privacy policies simultaneously for the data being collected (and the aggregation / filtering is done at the DV_i level). For example, one rule could state that WiFi is not to be shared. Another one could state that location should be shared, only under particular circumstances. This situation is represented in Figure 3.

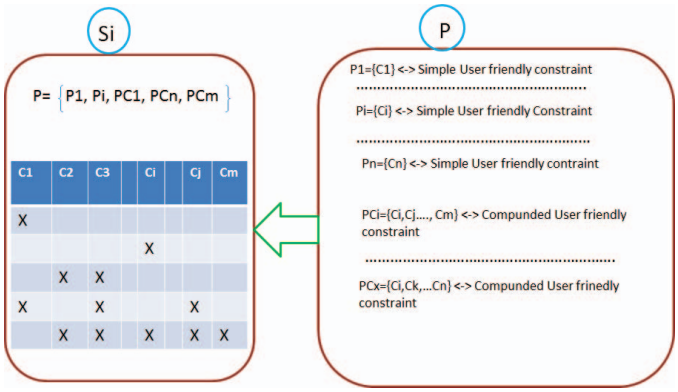

Fig. 3. Representation of how different security policies are applied at the Data Vault level.

V. Security Policy Formal Validation

Security policies in set P have the purpose of protecting sensitive information; the security policies allow or forbid information sharing maintained by sources S_i in data vault DV_i. Thus, for the formal validation, here we consider the simplified case where a data field is either shared or not. The demonstration for the general case of applying sharing operators is more complex, but the logic is similar, and we skip it due to lack of space.

Figure 4 represents how a source S_i shares information with other sources S_j or applications A_i.

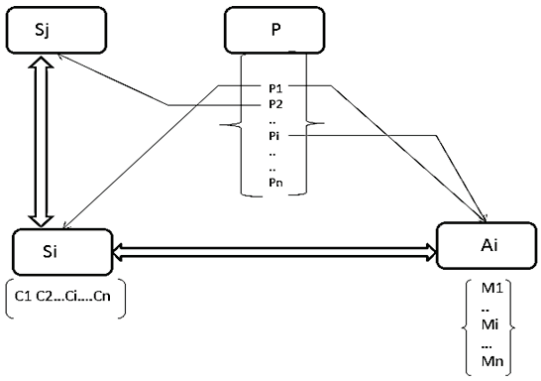

Fig. 4. The sharing process, influenced by security policies. Source S_i collects a number of fields C_i. The platform P manages a number of security policies P_i. The red arrows show that each sharing action is controlled, at the platform level, by security policies. Application A_i uses the data in a number of methods M_i. The green and blue arrow represent the logical flow of sharing operations (data can be shared with other sources, or with applications and services).

Formally, source S_i collects information, defined as:
$C = \{C_1, C_2, \ldots, C_i, \ldots, C_k, \ldots, C_n\}$
, where C_i represents particular data fields (information) collected.

The generic representation of a security policy P_i is:
$P_i = \{C_1, \ldots, C_m\}$
, where C_i represents a particular restriction regarding the sharing of information field C_i.

Let us consider that a source applies a policy $P_i = \{C_1, C_3\}$ and shares data $C = \{C_1, C_2, C_3\}$. This means that only field C_2 will be shared, because the other two are part of the no-sharing policy defined by P_i; or, formally, the platform will not share with other entities fields $\{C_1, C_2\} = C \bigcap P_i$.

All security policies applied by sources in set P are propagated to the data vaults DV_i. A source S_i can choose to adhere to one, more or no security policy P_i (Figure 3). Applying a security policy P_i implies creating a logical filter regarding the sensitive information that source S_i will not share with other entities in ITS.

Sharing information between entities can be divided in two cases: Source S_i shares data with (1) application or service A_i, or with (2) source S_j. The second case happens, for example, when the first source collects location data, and we have another source that is able to aggregate this and deliver (sensed) traffic information. Or, in a social context, two socially-connected nodes could be interested in sharing their collected data (sharing the wearabout of each others).

In the following theoretical demonstration, we prove that the proposed model is focused on the user privacy concerns, allowing source S_i to specify who and under what circumstances it can share data.

A. Sharing between source and application

Application A_i exposes a public interface composed of the following invocations/methods:
$M = \{M_1, M_2, ..., M_i, ..., M_m\}$
Method M_i can accept information according to one, more or no security policies described in P.

We refer first to case when method M_i is developed using the restricted information defined in security policy P_k.

A method M_i needs during execution a series of data fields:
$M_i\_C = \{C_i, C_j, C_m\}$

Let us assume the existence of security policy P_k that defines restrictions on exactly these fields $\{C_i, C_j, C_m\}$. We also assume several security policies (P_1, P_4, P_m) that do not impose access restrictions on any of the fields $\{C_i, C_j, C_m\}$.

A source S_i shares data according to a set of security policies:

- $S_i\_P = \{P_1, P_4, P_k, P_n\}$ (or, more generally, a number of policies including P_k). Sensitive data that source S_i will not share with method M_i includes in this case $M_i\_P \bigcap S_i\_P = \{C_i, C_j, C_m\} = M_i\_C$.
  Any other information field that is not part of the intersection can be shared with method M_i. We can notice that the intersection equals the set M_i_C, meaning the data needed by method M_i, but which will not be shared because is protected by the security policies accepted in DV_i.
- $S_i\_P = \{P_1, P_4, P_m\}$ (or, it does not include P_k.) Sensitive data that source S_i refuses to shares with method M_i is $M_i\_P \bigcap S_i\_P = \emptyset$.
  As the intersection is the empty set, source S_i can share all data with method M_i, as no security policy accepted by DV_i is broken.
- $S_i\_P = \emptyset$, which happens if source S_i did not place any constraints regarding its data (no information field is considered sensitive, as source S_i did not adhere to any security policy). In this case, all data can be shared with method M_i.

In all three cases, the sharing constrains defined by the user are never broken.

Let us consider now that method M_i needs restricted information affected by the set of security policies: $\{P_k, P_j, P_n\}$, where:
$P_k = \{C_k, C_m\}$
$P_j = \{C_j, C_l\}$
$P_n = \{C_n, C_p\}$
In this case, the method M_i cannot gain access to any of the fields $P_k \bigcup P_j \bigcup P_n = \{C_k, C_m, C_j, C_l, C_n, C_p\}$. Again, we can see that in this case, no matter what other security policies are in place, the method will never be able to access fields restricted by any of the individual security policies P_k, P_j, P_n (demonstration is similar to the previous one, considering that the intersection will not include additional elements than the ones already included by the reuniunion).

Another case is when method M_i needs data fields that are not part of any security policy defined in P (the case of backward compatibility, or how already existing applications are affected by newly introduced security policies).

Source S_i shares data according to the following set of security policies:

- $S_i\_P = \{P_1, P_4, P_k, P_n\}$
  Sensitive data that source S_i will not share with method M_i are defined by the reunion of fields affected by the policies in S_i_P.
- $S_i\_P = \emptyset$
  Source S_i did not place any constraints for its shared data (no information field is considered sensitive, as source S_i did not adhere to any security policy). In this case, it can share all data with method M_i.

In this case, again, we can see that no security policy is broken by the application of any constraint, defined by other security policy, at the platform or Data Vault level.

B. Sharing data between different sources

Source S_j needs to obtain information from source S_i, under the restrictions defined by security policy P_k. Source S_j needs to access the fields defined as:
$S_i\_C = \{C_i, C_j, C_m\}$
Source S_i shares data according to a set of security policies (cases similar to the ones above). First, we have the case:
$S_i\_P = \{P_1, P_4, P_k, P_n\}$

Sensitive data that source S_i will not share with source S_j is given by:

$S_i\_C \bigcap S_i\_P = \{C_i, C_j, C_m\}$ (again, we consider that policy P_k dictates a no-sharing policy for fields $\{C_i, C_j, C_k\}$.

Any other information field that is not part of the intersection can be shared with source S_j. We can notice that the intersection equals the set P_k, which means the data requested by source S_j is not shared by source S_i as it is protected by the security policies accepted in DV_i.

If $S_i\_P = \{P_1, P_4, P_m\}$, sensitive data that source S_i will not share with source S_j is given by

$M_i\_P \bigcap S_i\_P = \emptyset$.

As the intersection is null, source S_i can share its data with source S_j, as no security policy accepted by DV_i is broken.

When $S_i\_P = \emptyset$, source S_i does not place any restrictions regarding its data (no information field is considered sensitive, as source S_i did not adhere to any security policy). Thus, in this case it can share its data with source S_j.

Considering these three cases, we see that the proposed model is focused on the user privacy concerns, allowing the source to specify who and under what circumstances can share sensitive data.

VI. Experimental validation and results

Commercial ITS applications, like Google Traffic (www.google.com/maps) and Waze (www.waze.com), are designed to route people in the city. They rely on data collected by people running apps on mobile devices. However, even if users are collecting personal and potentially sensitive data, the way this data is used is not transparent. Users know little about how such applications (and the platforms behind) are using their data, who has access to it and under what circumstances.

Open-source routing engines like Open Source Routing Machine (OSRM) [9] provide an alternative in the open-source world. The advantage is that users can see the code behind, and therefore can form an opinion for the privacy mechanisms in place. However, because such engines are developed by volunteer developers, their features are not really that advanced. For example, OSRM routes traffic considering only the speed limits for various street segments (it does not include support for updating traffic conditions using data collected by drivers).

The reality is that, today, building ITS services (e.g., for navigation inside the city) is, at a certain degree, an elitist's domain: most of the good technologies are proprietary, and tailored to the actual investment. These services also require a costly (in many cases even a brand new) infrastructure, or some kind of interdependence with other services. All this can hinder the appearance and survival of new players and novel ITS service ideas. Recent research reports put a strong emphasis on the introduction of new types of ITS services (e.g. traffic information, route planning) relying on information coming from urban mobility [10]. Unfortunately, these are isolated and closed systems with solutions customized to a specific problem area; thus, they could face a difficult start up and are easily fated to unpopularity with a moderate and hard to keep user base.

The MobiWay project eliminates such constraints, by acting as a hub for connection and optimal support of ITS partners, supporting information management on a large scale. The project will generate new possibilities of extra value and improvement for already existing ITS solution providers, by allowing them to share mobility information and ITS services. MobiWay brings support for novel service ideas and initial development efforts, by boosting up collaboration with mobility users and fostering participation in the ecosystem of services. The project is expected to increase the introduction and adoption of Smart City services, and to directly involve citizens in the co-creation, exploration and benefits of these.

MobiWay targets the development of a robust, open-service and standards-based collaboration platform that will ensure interoperability between various sensing mobile devices and a wide-range of mobility applications. Its goal is to provide a framework of ecosystems, which allows to engage urban communities in exploiting the shared value of mobility that can be leveraged beyond the classical views of social networks, respectively the current trends of service creation.

In order to facilitate collaboration, the framework aims to distribute the different sources of information (coming from end users or services) within a service cloud composed from application providers. As various types of sensor data will be provided by mobility users, intelligent data gathering, processing and sharing have to be guaranteed between a mesh of hundreds of thousands of participants and the co-existing services with diverse interests. The project builds an ecosystem of integrated services and applications, flexible and reconfigurable, offered under multiple packages offered to users. The proposed MobiWay architecture is presented in Figure 5.

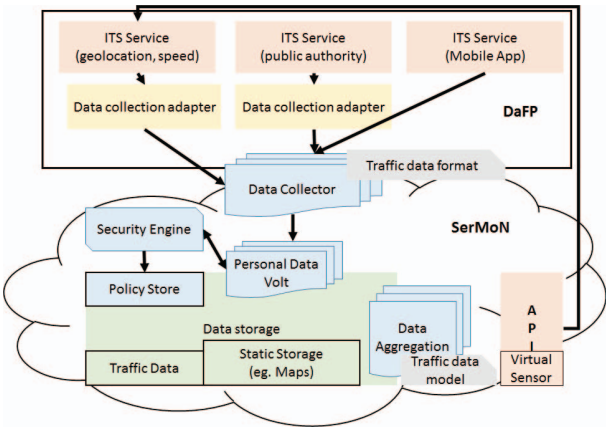

Fig. 5. MobiWay concept and architecture.

To assure extendibility, all the information related to mobility (position, time, speed, heading, etc.) or derived from mobile sensing and querying is separated from other application specific communication by channelling into a sensor data plane and exchanging through the Data Forwarding Plane (DaFP) system. DaFP aims to act as a grouping and forwarding

plane, responsible for interlinking and routing the proper urban mobility sources to subscribed services.

The new service demands can be satisfied by using dynamic information gathering mechanisms on modern smartphones. This affects their sensing, computing and communication stages and is provided by a middleware layer with capabilities to define Virtual Sensors (ViSe) and programmable sensor queries. This is the mechanism designed to share mobility and environment related information with other services.

The Service Mediating and Monitoring System (SerMoN) is where everything connects together. SerMoN defines a global view of the MobiWays ecosystem, and interconnects the DaFP, and controls the higher level configuration and data gathering/sharing aspects of the forwarding plane. Thus, the main role of SerMoN is to facilitate the introduction and maintenance of services, achieved through mediation, service specific selection and composition, by using the semantic knowledge collected from the ViSe capabilities, from DaFP control information and the registered services. The new services can also acquire information in order to evaluate the Quality of Service they can get from the system.

In MobiWay, after data is gathered, we store it in Personal Data Volts (PDVs), which are logically distinct data stores (i.e., per user or group of users) that declare and impose a set of policies or rules for each individual's data [5]. Before the data can be extracted and processed in MobiWay, the Security Engine (see Figure 5) is queried, and a decision is made about the availability of data with regard to that particular processing step. The Security Engine enforces the requirements of the PDVs. To achieve this, every processing request made by an ITS service triggers a policy check. Policies are further extracted on request from a Policy Store. The Security Engine is also responsible for preserving the privacy of all participants.

A pilot implementation for the proposed security model (and proposed privacy policy model) was developed by extending the SerMoN layer with components for routing over OpenStreetMap data [11]. We used open-source solutions like pgRouting [12], extended with own routing schemes that use data coming from traffic, and OpenStreetMap (for the street-level data). Real-time data is provided by users that run the MobiWay app (that collects timestamp, GPS location, and speed). SerMoN is developed as a scalable platform capable to store and process a large number of user supplied data per second [13]. We use private Data Vaults to restrict the publication of potentially sensible data. Each user has the ability to filter the amount of information he wants to share.

Thus, Data Security ensures that all information published in the SerMoN platform is sent according to what the user agrees. To define policies, we use specific structures in the internal database (see Figure 6). In particular, two tables are of interest: *policy*, and *user_policy*.

The *policy* table describes an individual policy. A policy is identified by its name and *app_id*. The third column contains a user-friendly description of the policy. Even though, at a first glance, it would seem that the *app_id* differentiator is not needed, we argue that a traffic collection infrastructure

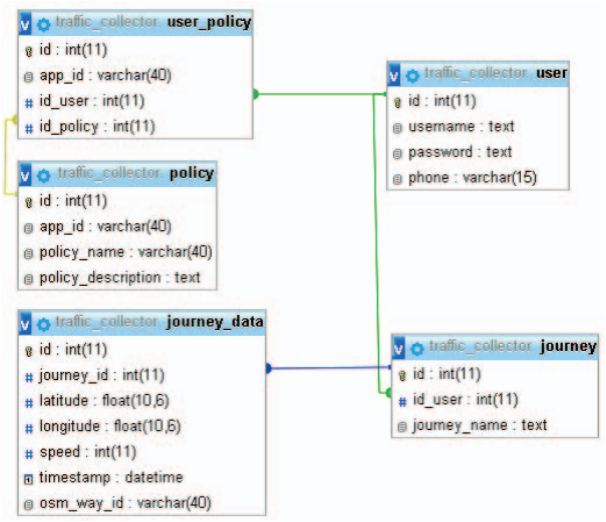

Fig. 6. The database structure for storing security policies.

may have more than one application running at the same time, with the same policy having different semantics, depending on the application. For example, let us consider two applications: *Routing* and *Social*, both running simultaneously, and each defining a policy named *Share Location*. For the routing application, the policy could mean the fact that the user's location could not be used for routing purposes, while for the social application, the policy could mean that the user specify how he should be visible to his friends on the map.

The *user_policy* table aims to establish a correspondence between a user and the policies that he agrees upon. As specified before, we consider a fine-grained approach to the policy problem. Each sharing capability is specified separately, and can be agreed / disagreed upon. We are going to consider that the presence of a tuple in the *user_policy* table means that the policy is accepted.

The platform includes a default set of policies:

```
1 mysql> select * from policy;
2 +----+--------+-----------------+-----------------------+
3 |id  |app_id  |policy_name      |policy_description     |
4 +----+--------+-----------------+-----------------------+
5 |1   |Mobiway|Share Location|User shares location|
6 |2   |Mobiway|Share Speed    |User shares speed    |
7 +----+--------+-----------------+-----------------------+
```

The policy enforcement is done in software, both on the client-side, and on platform-side. Data Security is concerned with the way that data published by the user is used, and is in charge with verifying that all policies are followed. To validate that sharing of data being published complies with any policy that the user accepts, we designed several experiments. We selected different policy options, and evaluated the sharing of that data.

For example, we considered two policy sets: *Share Location*, that specifying that we want to share GPS location, and *Share Speed*, with an intuitive effect.

For validating that we correctly publish the data according to user's security policies, we inspected the contents of the

database created in the pilot navigation Android application, with the help of an SQL query.

Share Location = True, Share Speed = True

```
mysql> select * from journey_data;
+----+-------------+-------------+-------+-------------+
| id | latitude    | longitude   | speed | osm_way_id  |
+----+-------------+-------------+-------+-------------+
| 15 | 44.463276   | 26.069309   | 52    | 23621824    |
+----+-------------+-------------+-------+-------------+
```

We can notice that both location and speed are correctly shared between the users and the targeted ITS service.

Share Location = True, Share Speed = False

```
mysql> select * from journey_data;
+----+-------------+-------------+-------+-------------+
| id | latitude    | longitude   | speed | osm_way_id  |
+----+-------------+-------------+-------+-------------+
| 16 | 44.463287   | 26.069313   | NULL  | 23621824    |
+----+-------------+-------------+-------+-------------+
```

Also, for the next case, only location is shared. In this case, speed is a sensitive data that is not shared with the targeted ITS service.

Share Location = False, Share Speed = True

```
mysql> select * from journey_data;
+----+-------------+-------------+-------+-------------+
| id | latitude    | longitude   | speed | osm_way_id  |
+----+-------------+-------------+-------+-------------+
| 11 | NULL        | NULL        | 52    | NULL        |
+----+-------------+-------------+-------+-------------+
```

Now, we consider the case where the user wants to share only his speed. Therefore, location is sensitive data that is not shared with the targeted ITS service.

Share Location = False, Share Speed = False

```
mysql> select * from journey_data;
+----+-------------+-------------+-------+-------------+
| id |latitude     | longitude   | speed | osm_way_id  |
+----+-------------+-------------+-------+-------------+
| 16 | NULL        | NULL        | NULL  | NULL        |
+----+-------------+-------------+-------+-------------+
```

In all these cases, the user is not sharing either location or speed, according to its security policy.

VII. Conclusions

Intelligent Transportation Systems (ITS) attempt to relieve congestion and decrease travel time by assisting drivers on making informed decisions while driving, by selecting better routes, departure times, and even various modes of travel. Most current ITS platforms, unfortunately, fail to include adequate support for dealing with the privacy requirements coming from users voluntarily participating. The data being collected, if aggregated over periods of time, could disclose sensitive information to third-party applications about preferences in routes, or to malicious users about critical traffic events.

In this, we propose a new model for the definition of security policies targeted specifically at the ITS. Our approach ensures that have the possibility to control access to sensitive data before is is shared with various ITS applications and services. On the same time, we want users to be able to easily adapt and modify their security policies at any given time.

We identified, in this respect, several requirements: *Data security* means data access must be secure and controlled only by the owner of the information (source S_i). However, the security mechanism must be flexible enough in order to allow security policy changes in accordance to source S_i context situation. *Transparency in design* means that privacy requirements must be integrated in the development process for ITS services / applications. Finally, security should be focused on user privacy needs regarding data usage. Sharing data must be in accordance to user privacy needs, these criteria being used when developing a ITS service, not vice-versa.

Using the security model described, we allow sources S_i to protect their sensitive information before sharing it with ITS service, we facilitate the development of ITS services so that they integrate in an efficient way user privacy requirements, and we ensure transparency in using and processing information exchanged between sources S_i and applications and services A_i.

Acknowledgment

The research presented in this paper is supported by national project MobiWay, Project PN-II-PT-PCCA-2013-4-0321, and by COST Action IC1303: Algorithms, Architectures and Platforms for Enhanced Living Environments (AAPELE).

References

[1] N. D. Lane, S. B. Eisenman, M. Musolesi, E. Miluzzo, and A. T. Campbell, "Urban sensing systems: opportunistic or participatory?" in *Proceedings of the 9th workshop on Mobile computing systems and applications*. ACM, 2008, pp. 11–16.

[2] I. Krontiris and T. Dimitriou, "A platform for privacy protection of data requesters and data providers in mobile sensing," *Computer Communications*, 2015.

[3] ——, "Privacy-respecting discovery of data providers in crowd-sensing applications," in *Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference on*. IEEE, 2013, pp. 249–257.

[4] K. Shilton, "Four billion little brothers?: Privacy, mobile phones, and ubiquitous data collection," *Communications of the ACM*, vol. 52, no. 11, pp. 48–53, 2009.

[5] M. Mun, S. Hao, N. Mishra, K. Shilton, J. Burke, D. Estrin, M. Hansen, and R. Govindan, "Personal data vaults: a locus of control for personal data streams," in *Proceedings of the 6th International COnference*. ACM, 2010, p. 17.

[6] N. T. Siebel and S. Maybank, "The advisor visual surveillance system," in *ECCV 2004 workshop applications of computer vision (ACV)*, vol. 1. Citeseer, 2004.

[7] B. Bonne, A. Barzan, P. Quax, and W. Lamotte, "Wifipi: Involuntary tracking of visitors at mass events," in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*. IEEE, 2013, pp. 1–6.

[8] Y.-K. Wang, "Context awareness and adaptation in mobile learning," in *Wireless and Mobile Technologies in Education, 2004. Proceedings. The 2nd IEEE International Workshop on*. IEEE, 2004, pp. 154–158.

[9] OSRM, "Open source routing machine," http://project-osrm.org/, accessed on December 20, 2015, 2015.

[10] R. Payre, "The importance of being connected. city networks and urban government: Lyon and eurocities (1990–2005)," *International Journal of Urban and Regional Research*, vol. 34, no. 2, pp. 260–280, 2010.

[11] OSM, "Openstreetmap project," https://www.openstreetmap.org, accessed on December 20, 2015, 2015.

[12] pgRouting, "pgrouting project," http://pgrouting.org, accessed on December 20, 2015, 2015.

[13] C. Fratila, C. Dobre, F. Pop, and V. Cristea, "A transportation control system for urban environments," in *Emerging Intelligent Data and Web Technologies (EIDWT), 2012 Third International Conference on*. IEEE, 2012, pp. 117–124.