Available online at www.sciencedirect.com
ScienceDirect
Computers & Security 00 (2017) 1–18
Computers & Security

Cloud-of-Things meets Mobility-as-a-Service: an Insider Threat Perspective

Franco Callegati[a], Saverio Giallorenzo[a,b], Andrea Melis[a], Marco Prandini[a]

[a]*Università di Bologna, Via Zamboni, 33, 40126 Bologna, Italy*
[b]*INRIA, France*

Abstract

Mobility as a Service (MaaS) applies the everything-as-a-service paradigm of Cloud Computing to transportation: a MaaS provider offers to its users the dynamic composition of solutions of different travel agencies into a single, consistent interface. Traditionally, transits and data on mobility belong to a scattered plethora of operators. Thus, we argue that the economic model of MaaS is that of federations of providers, each trading its resources to coordinate multi-modal solutions for mobility. Such flexibility comes with many security and privacy concerns, of which insider threat is one of the most prominent. In this paper, we revise and extend previous work where we classified the potential threats of individual operators and markets of federated MaaS providers, proposing appropriate countermeasures to mitigate the problems. In addition, we consider the emerging case of Clouds of Things (CoT) for mobility, i.e., networks of ubiquitous, pervasive devices that provide real-time data on objects and people. Automation and pervasiveness make of CoT an additional attack surface for insiders. In an effort to limit such phenomenon, we present an overlay networking architecture, based on gossip protocols, that lets users share information on mobility with each other. A peculiarity of the architecture is that it both constraints the quality and quantity of data obtainable by insiders, optimizing the routing of requests to involve only users that are able answer them.

Keywords: Mobility-as-a-Service, Federated Platforms, Insider threat, Cloud-of-Things, Internet-of-Things

1. Introduction

The term Cloud Computing denotes a dynamic infrastructure where users access services without regard to where the services are hosted [8]. The concept of Mobility as a Service (MaaS) [57] takes inspiration from such a model and brings it into the context of transportation. In Cloud Computing, the architecture that runs the services is dynamic and transparent to users. Likewise, MaaS hides a dynamic composition of solutions provided by different travel agencies behind a consistent interface. Hence, MaaS users experience traveling over complex itineraries as if they were provided by a single agency.

Due to regulatory and logistic issues, mobility resources are administrated and owned by a scattered plethora of mobility operators (traditional travel agencies and providers of data for mobility). Thus, we argue that the leading economic model of MaaS markets is that of federations of mobility operators, each trading its resources. In such a federated market, operators can dynamically partner with each other, still preserving their individual autonomy and without the need for a centralized regulation authority. On these premises, we are currently developing a Service-Oriented platform, called *Smart Mobility for All*[1] (SMAll), built on the concept of federated Cloud Computing [58, 9] and purposed to support liquid markets for transportation.

During the development of SMAll and through the collaboration with our industrial partners (public administrations, local travel agencies, etc.), we identified and analyzed many security issues spanning from a single operator to a federation of operators. In this context, we deem malicious insider activity one of the most prominent threats, spanning from standard threats against cloud installations [42] to insider issues specific to the contexts of mobility and of markets of services.

Motivation. Fig. 1 depicts a cross section of an instantiation of SMAll, where the colored entities outside of the boundaries of SMAll (bordered with double lines) are public transportation agencies, private companies, on-line communities, and

[1] https://github.com/small-dev/SMAll.Wiki/wiki

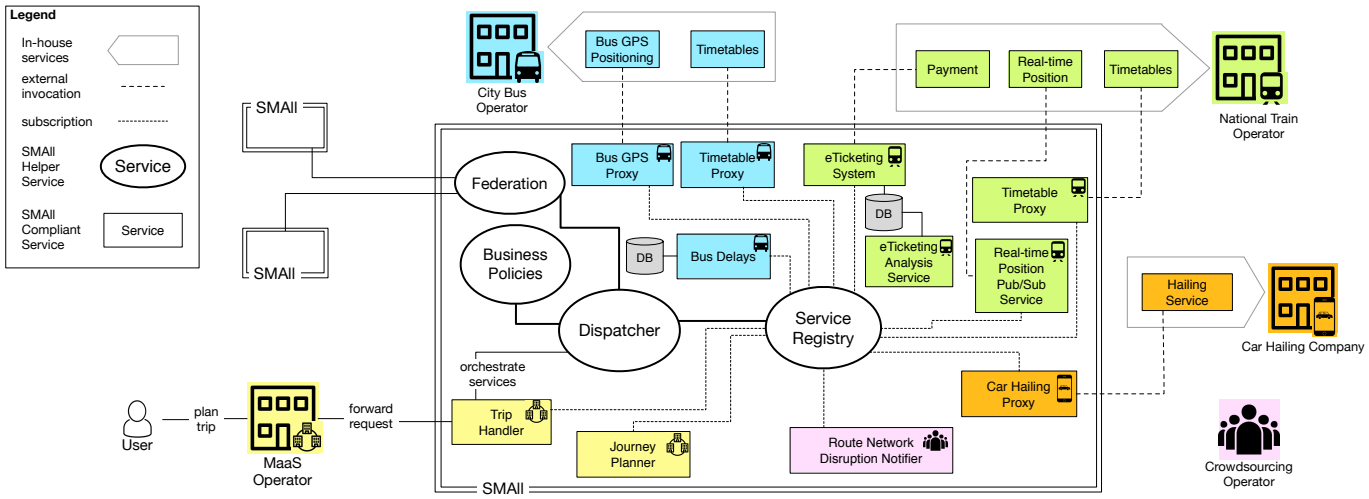

Figure 1. Example of the SMAll architecture.

MaaS operators.

Even when considered in isolation, the agents in the platform already entail well-known threats due to insider activity. For example, the City Bus Operator represents a threat to the privacy of drivers since GPS positioning can reveal sensitive information on their conduct, which is forbidden under some legislation; however, also drivers represent an insider threat to the Bus Operator: they can disable the GPS device on their vehicles, compromising the reliability of the GPS positioning system and that of the other services that depend on it[2] (e.g., the Bus Delays service that estimates bus arrivals based on vehicle GPS positions). Finally employees can manipulate the services and their data, damaging the company by extracting restricted information or causing outages.

Broadening our scope to federated interactions, we focus on the MaaS Operator in Fig. 1 that, for example, deploys a Journey Planner service for providing dynamic multi-modal trips to users. The service orchestrates other federated services in SMAll: it uses information on scheduling, availability, disruptions, and the position of buses, trains, and on-demand cars. As expected, the threats highlighted for single operators surface (and possibly combine into new ones) to higher-level federated scenarios. Consider the case in which the City Bus Operator allows the MaaS Operator to access the Bus GPS Proxy service. With the raw data on the real-time position of buses, the MaaS Operator can undertake many malicious activities to the detriment of the Bus Operator, e.g., passing relevant information to its competitors. Another important threat comes from the extraction of sensitive information from aggregated/anonymized data. Consider the case of a Bus Operator, which is aware of the threat posed by the Bus GPS Proxy service, and thus it decides to markets only its Bus Delays service. This countermeasure could be ineffective, since also aggregated data like the temporal approximation of the arrival of buses might let the MaaS Operator extract [75] the actual position of vehicles (possibly optimizing the accuracy of the extraction [45]).

Contribution. As exemplified, in the context of MaaS operators, the definition of what an insider is can assume subtle nuances depending on the considered scenario. In this work, guided by our experience with the development of SMAll, we describe the security issues concerning insiders within such a federated market of services. In doing so, we consider two distinct perspectives: *i*) the one of high-level services traded in an open platform and *ii*) the one of low-level data sources for mobility, focusing on the emerging case of Clouds-of-Things [5, 38] for mobility. Our contributions are organized as follows.

Insider Threats in MaaS Markets. We consider the high-level perspective of services for mobility in MaaS markets. Doing so, we integrate and extends material from [11], where we introduced the case of MaaS markets and analyzed the possible emerging insider threats. Here, we present a revised version of the proposed analysis. To structure our exposition, we follow a tiered view of the MaaS markets called the MaaS Stack, presented in depth in [30]. In § 2 we give a brief account of the MaaS Stack. Then, in § 3 we discuss our findings: we consider each tier of the MaaS Stack, we define what an insider is for each of them, we analyze the related threats, and we describe the possible countermeasures.

Cloud-of-Things for Mobility: Insider Threats. We analyze insider threats in the context Cloud-of-Things (CoT) for MaaS and we propose an architecture that constraints the quality and quantity of data that an insider could obtain from users. In doing so, the architecture also optimizes the routing of requests only to those users that are able to answer them. To achieve these results, the proposed architecture creates an overlay network among users, so that requests are bound to limited lo-

[2] The issues are far from being just speculative, as we actually encountered them collaborating with one of our industrial partners.

calities, guided by the capabilities (knowledge, proximity, etc.) declared by users, and spread following a gossip-based protocol [34, 7]. In § 4 we briefly illustrate the relationship between Mobility-as-a-Service and Cloud-of-Things, the latter being an increasing source of real-time data for mobility services. Since CoTs represent another attack surface for insiders, we analyze these threats in § 4.1. From our analysis, in § 4.2 we discuss how an overlay network of CoTs can mitigate some of the identified threats. Then, in § 4.3 we give a brief account on gossip-based networks and describe the how our overlay network employs gossip-based propagation of information to achieve both optimization of queries and locality of data. In § 4.4 we present our overlay network. Finally, in § 4.5 we report a thorough analysis on how and to which degree our overlay network mitigates some of the threats of MaaS, identified § 3, proposing possible future evolutions.

2. The MaaS Stack: An Overview

In this section we briefly overview the MaaS Stack (Fig. 2), a structured view that we assembled to guide the development of SMAll. In § 3 we use the MaaS Stack to analyze the insider threats of each tier.

Tier I | eMobility Operators. The first tier of the MaaS Stack is that of *eMobility Operators*. An eMobility Operator is an entity that owns, administrates, and exposes software functionalities regarding mobility, provided in a machine-readable form. In tier I of the MaaS Stack, eMobility Operators are considered in isolation (i.e., not using and integrating the services of other operators). For example, the National Train Operator represented in Fig. 1 is an eMobility Operator that owns services for purchasing tickets, accessing timetables, and receiving real-time position of vehicles. Cloud-of-Things represent one of the main information sources used by the services in this tier, in particular streaming real-time data on the position of vehicles and people, availability of parking spots, etc..

Tier II | Business Intelligence. The second tier of the MaaS Stack still focuses on single eMobility operators but it enriches the taxonomy of services with the category of *Business Intelligence*. These services are not meant for users but for eMobility operators; they span over first-tier services by monitoring and analyzing their usages. Business Intelligence services provide insight on the performances of eMobility operators. For example, the eTicketing Analysis Service (Fig.1) of the Train Operator can suggest to the latter new pricing policies as well as reporting rarely used routes that could be merged/discarded.

Tier III | MaaS Operators. The last tier of the MaaS Stack is that of MaaS Operators, i.e., eMobility operators that federate and integrate their services with those of other eMobility operators. Each MaaS operator provides to its users information and transit services of other operators as its own. The principle resembles that of "roaming" in GSM phone networks [52], where users connect through the services of another phone company when traveling outside the geographical coverage area of the home network. The MaaS Operator represented in Fig. 1 can federate with the National Train Operator and the City Bus Operator and it can offer multimodal journeys that span different means of transportation and have nation- to city-wide scopes. This example introduces the last fundamental element of the third tier of the MaaS Stack: *Clearing* services to account for federated usages and compensate operators according to the established Business Policies.

To support the mentioned features in SMAll, we are currently developing and integrating components to deploy services, to support the definition and enforcement of business and clearing policies, and to federate many instances of the platform. During the development, we recognized and investigated security issues derived from the openness of our federated platform. In the next section, we consider each tier of the MaaS Stack, we define what an insider is for each of them, we analyze the related threats, and we describe the possible amendments to counteract them.

3. Insider Threats in MaaS Scenarios

Statistically, insider threats are one of the most expensive security issues for business companies [53]. One prominent reason of these expensive outcomes is that companies did not foresee all possible malicious insider activities [71]. Indeed, the problem is not the lack of proper countermeasures as much as the difficulty of identifying a malicious insider in the first place. Literature abounds with guidelines and principles aimed at providing general descriptions of the context and the identity of the insiders [27, 15]. However, experts agree that the strong contextual variance of threats [64] makes providing a general yet precise identification of all possible insiders difficult.

Thus, we deem useful to share the experience we gained in the context of services for mobility (both at software and physical level). Moreover, our background on the development of SMAll provides insights on the possible threats deriving from federated cloud architectures, built for deploying, publishing, and trading services. Federated clouds have been already analyzed in literature [42, 55, 19], however we deem important to include the related threats in the frame of the emerging Mobility-as-a-Service scenario.

In § 3.1–3.3, we illustrate, for each tier of the MaaS Stack (cf. Fig. 2), the insiders, the related attacks, and the possible countermeasures, as found in the state of the art and as implemented in SMAll. In Fig. 3 we report a table that summarizes our findings. Agents and threats are classified according to the categories identified by Casey in [13] and the CERT technical report [67]. We dedicate the last paragraph of this Section to a brief description of the methodology we followed to recognize the threats and the respective countermeasures.

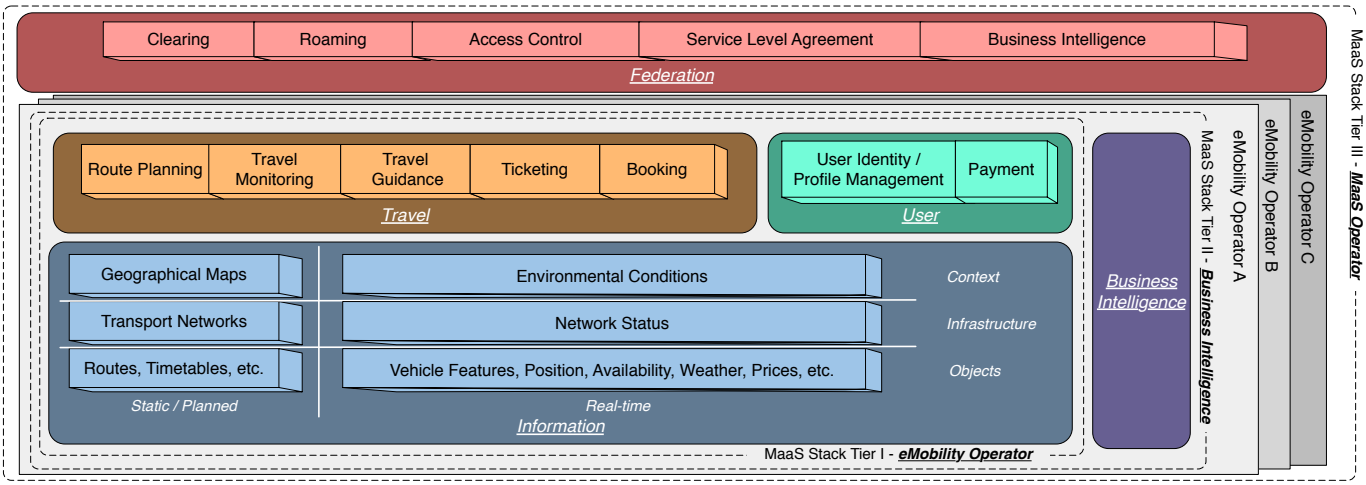

Figure 2. The MaaS Stack.

Methodology. As mentioned, adopting a narrow definition of insider may hinder the identification of threats specific to particular contexts. Therefore, in our investigation, we prefer to look at insiders from a general point of view [6]:

> *A trusted entity that is given the power to violate one or more rules in a given security policy [. . .] the insider threat occurs when a trusted entity abuses that power.*

This definition hints that an insider is determined by the role played as member of a system and related to the deployed control rules and the pursuable malicious goal(s). In our context, the most classic scenario is one where the insider is within the service of the victim, e.g., a programmer that manipulates the behavior and the data of a service. However, orchestrations spanning many providers, hallmark of the SMAll platform, lead to subtle yet relevant threats. Consider the case of federated partners. On one hand, the provider of a service exposes itself to threats posed by members that use its service — security issues span from misuse of information extracted from the service to over-usages that entail unforeseen costs or outages — on the other hand, an agent that orchestrates services of other partners is a man-in-the-middle able to leak private information, counterfeit data or use its vantage point to extract strategic patterns from partners.

Regarding countermeasures, we structured our analysis of the possible alternatives following the review compiled by Hunker and Probst [40], encompassing the three approaches: *i*) *Prevention*, comprising the definition of strong access control rules, data management systems (including data masking and data camouflage), and mechanisms to guarantee data provenance and data trustworthiness; *ii*) *Detection*, that usually goes hand-in-hand with dissuasion mechanisms such as techniques of data management and service invocation that make abuses extremely expensive in terms of computing power; *iii*) *Mitigation*, that exploits auditing and monitoring techniques, often based on machine learning, to automatically react to insiders' activities.

3.1. MaaS Stack | Tier I

As reported in § 2, the first tier of the MaaS Stack focuses on single eMobility operators and categorizes their services. In this tier, the ecosystem of services has a flat structure and all members play the same role of providers, without any interaction between each other. Here, insiders can be pinpointed within two types: *i*) *users* authorized to interact with services and *ii*) *managers* (also seen as owners) of the services. In the reminder, we call Users the members of the first type and Managers the members of the second one. The distinction between the two types is trivial: while Users have limited access to data and functionalities of a service, Managers can have full or partial control (depending on the responsibility level) over the life-cycle of the service and its resources. Users allowed to interact with SMAll services can basically pose two types of threats: *i*) perform fake data injection (for crowdsourcing-based services) and *ii*) sharing the access to the services or to the respective data. Users can also exploit vulnerabilities to acquire Manager privileges (configuring an *Insider Impersonation* threat). However, we do not include a discussion on these kind of attacks as they coincide with those described for Managers. Regarding Managers, their main threats comprise:

- manipulation of the behavior of a service, i.e., the computations done by a service;
- manipulation of the workflow among services, i.e., the direction and sequence of information flow among services;
- stealing data, metadata, and performing malicious analyses;
- exposing sensitive information.

Following the first tier of the MaaS Stack, we describe the possible insider attacks of each category of services.

Tier	Agent	Agent Type	Insider Threat	Event Type
1 & 2	User	Competitor, Untrained/Distracted Insider, Outward Sympathizer	Fake Data Injection	Sabotage
			Service Behavior Manipulation	Product Alteration, Sabotage
	Unauthorized Guests	Competitor, Theft, Activist	Man-in-the-middle Attack	Misuse
	Developers	Competitor, Partner, Disgruntled Insider	User Impersonation	Sabotage, Espionage, Misuse
	Service Administrators	Partner, Disgruntled Insider, Untrained/Distracted Insider, Supplier	Insider Impersonation	Sabotage, Espionage, Misuse
	Service Managers	Partner, Disgruntled Insider, Untrained/Distracted Insider, Supplier	Crowdsourcing Attacks	Sabotage, Financial Fraud
	Agent after privilege escalation	Activist, Competitor	Data Leakage - Accidental - Data Theft - Resale of Data and Access - Business Intelligence Data Theft	IP Theft, Opportunistic Data Theft, Physical Theft, Accidental Leak
3	Federated MaaS Member	Competitors Nation State Partner Supplier	Data Manipulation, Trustability, Tampering of Data Provenance, Data Trustworthiness	Financial Fraud, Product Alteration
			Service Behavior Manipulation	Financial Fraud, Product Alteration
			Composition of Unverified Services and Data	Misuse, Sabotage, Espionage, Product Alteration
			Denial of Service	Sabotage
			Service Workflow Manipulation	Misuse, Sabotage, Espionage, Product Alteration
	MaaS Competitor	Nation State, Partner, Supplier	Data Analysis: - Pattern Extraction - Data Mining - Data Exploitation trough data crossing	Accidental Leak, Opportunistic Data Theft Espionage, Financial Fraud
	Helper Service	Competitors, Nation State, Partner, Supplier		

Figure 3. Summary table relating the tiers of the MaaS Stack to their concerning insider threats.

3.1.1. Information

The category of Information spans from basic services that publish raw data (e.g., timetables or the position of vehicles) to higher-level services that elaborate raw data to extract new information (e.g., the expected delay of buses whose calculation requires the position of a vehicle and its scheduled plan). As already mentioned, the Information category is the point of conjunction between high-level services for MaaS — and, by extension, MaaS markets — and sensing devices in Cloud of Things. Indeed, most of the real-time data used by this services is generated by embedded, portable or even wearable [26] devices/sensors. Since in this work we separate the analysis on services and on CoTs, here we consider only threats at the level of services, dedicating § 4.1 to the analysis of threats on CoTs.

Notably, since Information services orchestrate other services to calculate and publish these refined data, they are subject to *Service Workflow Manipulation* and *Composition of Unverified Services and Data* threats. We omit to present these issues in this Section and defer the discussion to § 3.1.2.

Data Leakage. Data leakage is the accidental distribution of private or sensitive data to unauthorized entities [65, 12]. In SMAll, both Users and Managers can cause data leakage. Users can share data to other, non-authorized Users. Similarly, Users can also share their access to services, which could lead to data leakage but also to other type of threats like *User Impersonation*. As expected, data leakage becomes even more serious when considered for Managers that can share or steal sources unreachable by users.

Countermeasures. Data leakage poses a serious issue in open networks where the transition of data is not regulated nor monitored in their path. In these regards, SMAll holds a privileged position. In fact, all communications among the services in the platform happen through the Dispatcher (cf. Fig 1), which can log the quality and quantity of information required by all Users. Obviously, this guarantee ceases when data exits the platform. The same tracing system applies also to Managers.

Crowdsourcing Attacks. Users can perform insider attacks on crowdsourcing services. These services handle data streamed from sensors and devices or through direct signaling of the users. An example is a crowdsourcing service where users can report architectural limits for people with disabilities [47]. In this case, insiders can feed the service with fabricated data to alter the normal behavior of services, e.g., by directing users through specific pathways.

Countermeasures. For the sake of completeness and clarity, let us start from the literature regarding "classic" threat scenarios. Cho et al. [14] examined how insider attacks can exploit security holes in a trusted network of sensor nodes. This work is of interest for our platform because it shows how even trust-based approaches, in architectures that have to unify many nodes, are not guaranteed to prevent attacks.

In [28], the authors described how access control policies for a database management system can be exploited by insiders when the control restrictions to be enforced may come from different authorities. Shatnawi et al. [66] made a similar analysis but based on the detection of malicious usage of a data source, which is equivalent to our case of a malicious influence of data source services exposed by the SMAll platform.

An interesting work that can be applied to our architecture is [70]. Here the authors implemented a pool of honeypots to catch insiders. A honeypot is an information system resource whose value lies in unauthorized or illicit use of that resource. The high flexibility of honeypots — able to play a huge variety of SMAll-compliant services — is essential to make insiders expose themselves. Another useful method that can be easily built within SMAll is a reporting system for crowdsensing and crowdsourced data, implemented in [46]. The reporting system is based on the mapping of what the authors called Point of Interest (POI). Each POI and its related data can be added to the system by means of one or

more reports. Reports are classified in three different source classes, accordingly to the reputation of the user that collects the data.

Service Behavior and Data Manipulation. As expected, insider threats posed by Managers constitute a more complex scenario. This type of insiders can access and modify the raw data of services as well as manipulating their logic to present altered results. Notably, since in our context the physical world mixes with that of software services, we extend the role of Managers not only to the developers that can modify the actual code of the service but also to conductors and other operators: agents that can access and manipulate the physical devices that feed the services.

The manipulation of these services can have many purposes from the point of view of an insider. For example, during the development of SMAll we interacted with many industrial partners, among which there were some public transportation companies that provided real-time positioning of their vehicles. However, some of these companies did not report the actual position of buses and instead published fake positions to mirror the exact planned schedule. In another case the service worked intermittently. In the first case, the company provided fake data to protect itself against possible penalties due to delays, in the second case the positioning service went down for certain rides due to drivers that disabled the in-vehicle positioning devices either for fraudulent purposes (to avoid being scrutinized) or even for shallow reasons such as to disable annoying automatic voice announcements.

Countermeasures. Interesting works tackle the issue of how to predict insiders activities. Ho et. al. [37] implemented a detection mechanism for single users based on analyses of changes on the writing style of the user after an attack occurred, using machine learning algorithms. Althebyan [1] implemented a prediction model based on graph theory approaches, to push alert once a detection risk mechanism finds that users are performing actions that might lead to compromise the system services.

Studies also exist aimed at discovering malicious command execution. Among the most relevant works, Kamra et al. [41] and Mathew et al. [44] focus on the analysis of anomalous commands executed on databases. In particular, they proposed a syntax analysis system to detect anomalous queries; the former analyzed the submitted SQL queries, while the latter focused on data retrieved from queries. Doss and Tejay [20] conducted a similar investigation as a field study within an enterprise, where analysts were monitored while performing their jobs. Again, these results can be readily applied in our architecture, especially considering that tier I services will in any case be monitored by probes needed to build Business Intelligence services of the second tier.

In principle, the SMAll service deployment interface can verify the correctness of an application before accepting it. In practice, this operation is very hard to perform. One indicator of correctness is the compliance to a template of acceptable interfaces for the kind of service the application provides. However, it is very difficult to define templates strict enough to allow sensible compliance checks, but general enough to avoid hindering the deployment of legitimate services.

Another important detection strategy that we considered is to implement a mechanism that could guarantee, in every moment, a reproducibility of the results of a service. With provenance certifications of raw data and their propagation to results, it is possible to implement a reference monitor to verify compliance between results and expected values. In case of conflicts between the declared results and the actual ones, SMAll could discover what has been tampered with: the source data, or the service logic. This detection can also feed a data trustworthiness rating system.

Finally, another way to check correctness is to look at the actual behavior of the application, as it is common in anti-malware checks. These techniques are far from infallible, and their scope falls much shorter than what is required in our setting. Indeed, in this context a malicious behavior can be a subtle deviation from the correct calculation [51], which is far more difficult than the detection of traditional malicious behaviors (e.g., damaging or self-replicating ones). Promising techniques, which can benefit from the execution of all the services on the SMAll platform, are those based on the aggregation of multi-domain information [23, 2].

3.1.2. Travel

Services in the Travel category orchestrate Information ones to provide highly coordinated functionalities to users. Since the services in this category heavily rely on composition to provide their functionalities, their main concerns regard their workflow.

Service Workflow Manipulation. Managers can modify the expected flow of information among services for many purposes. As an example, consider the Manager of a service called Bus ETA that predicts bus arrivals. In its calculations, Bus ETA uses three source-services, respectively for traffic, GPS positioning, and weather forecasts. Although the Manager preserves the logic (i.e., the behavior) of the Bus ETA service, by simply changing the workflow, i.e., the bindings of the Bus ETA to the other services, she can make (some) of the sources unreachable, either completely disabling the Bus ETA service or modifying the resulting output due to missing data.

Countermeasures. SMAll already provide tools to contrast service workflow manipulations through the helper services Dispatcher and Business Policies (Fig 1). Indeed, when Managers deploy their services in SMAll, they also define the related access rules (stored and retrieved in the Business Policies service). Then, all workflow compositions pass through the Dispatcher service that logs them and enforces the established access policies. In this way, unexpected workflows are detected, logged, and (depending on the access rules) forbidden. The monitoring capabilities of the Dispatcher can also be enhanced with techniques like [63], where the authors propose an analysis to detect malicious workflows and [24],

that employs machine learning engines similar to the ones used in dynamic malware analysis to detect malicious workflows. Finally, based on service specifications, we can create workflow graphs for strategic mitigation [73].

Another promising approach comes from the field of Choreographic Programming [50]. The use of choreographies to implement workflows among services is relatively new [29]. We deem choreographies an effective prevention tool that lets partners agree on a formal definition of their workflows, which can be later compiled into their respective, compliant services. Moreover, in the dynamic context of SMAll, tools like [17] can aid partners in updating their agreed workflows even at runtime (i.e., without stopping their running services). These updates would be still conditioned to a general agreement and maintain the same guarantees of the original services.

Mitigation techniques can be also developed following e.g., [31]. The idea would be to develop a SMAll helper service that monitors workflows and, once an attack by an insider is discovered, it appropriately redirects the workflow to avoid further damage.

Composition of Unverified Services and Data. In the context of mobility, verified information is of paramount importance. However, in a service-oriented architecture, the tricky part to deal with is that a service invocation can be seen as a collection of workflows. These workflows can compose many levels of services, each processing and modifying the data before its final destination. These services inherently include the logic of the composed services and, by extension, also the possible manipulations executed by insiders. As an example, consider a journey planner that uses a real-time traffic report service to avoid traffic jams and roadblocks. Since the journey planner directly integrates the information from the traffic report service, manipulating information of the latter alters the solutions of the journey planner, diverting travelers towards certain pathways. This case presents an interesting nuance: the insider is not a direct Manager of the considered service (i.e., the journey planner), instead it is the Manager of a composed service (the traffic report) that twists its contribution to alter the behavior of the planner. In this context also trustability, provenance, and trustworthiness of data and/or services should be considered as possible targets of attacks. For example, tampering with data provenance is a source of attack [69] that in a MaaS scenario can see malicious operators claiming to publish genuine data of a competitor, actually forging them.

Interfering with the certification of data trustworthiness is another possible vector. In this case, it is very difficult to block attacks in which, e.g., the creator advertises a data source of given quality, but then exposes a degraded version to keep the advantage of more precise/timely information for herself. A related trustworthiness scenario is that of an insider who succeeds in registering a rogue service. For example, a modified travel planner could deflect routes to favor or damage certain businesses; a modified delay-checking application could hide or amplify violations of agreed service levels.

Countermeasures. A service must support the provision of different sources of data along with their associated metadata (e.g., used to verify their provenance). However, SMAll shall also provide techniques, embodied by helper services, to transform those data into verified information. There are different approaches that provide a solution to the problem of recognizing the source of a data stream. Literature agrees [32] that the requirements for a provenance management system are: *Verifiability:* a provenance system should be able to verify a process in terms of the actors (or services) involved, their actions, and their relationship with one another; *Accountability*: an actor (or service) should be held accountable for its actions in a process. Thus, a provenance system should record in a non-repudiable manner any provenance generated by a service; *Reproducibility*: a provenance system should be able to repeat a process and possibly reproduce a process from the provenance stored; *Preservation*: a provenance system should have the ability to maintain provenance information for an extended period of time. This is essential for applications run in an enterprise system; *Scalability*: given the large amounts of data that an enterprise system handles, a provenance system needs to be scalable; *Generality*: a provenance system should be able to record provenance from a variety of applications; *Customizability*: a provenance system should allow users to customize it by setting metadata such as time, events of recording, and the granularity of provenance.

In these regards, it would be useful to deploy technologies to certify the metadata related to a data stream and manage its validity during time and re-elaboration [72]. According to works like [68], this problem could be solved only with the creation of a public-private key system for data stream certification. A good reference is the system developed in [74], describing a cryptographic provenance verification approach for ensuring data properties and integrity for single hosts. Specifically, the authors designed and implemented an efficient cryptographic protocol that enforces keystroke integrity. This kind of protocols can be integrated as a helper service in SMAll. However, public-key schemes are known for their significant computational load, thus existing techniques may not be suitable for high-rate, high-volume data sources. Moreover, there could be the need for an algorithm for the provenance of composed data. In some cases, data originated from the composition of raw (or otherwise lower ranked) sources should be accompanied by suitable metadata for verifying the provenance of the input values, in a cryptographically strong way. In the context of SMAll, it could be important and useful to capture and understand the propagation of data.

The combination of metadata- with key-propagation management can guarantee a good level of trust in provenance management systems. Works in the direction of [35] discuss how to support provenance awareness in spatial data infrastructure and investigates key issues including provenance modeling, capturing, and sharing, useful to implement key propagation systems.

Finally, we address trustability, provenance, and trustworthiness of services and/or data.

Trustability needs to be measured by indicators for data quality and service behavior. Values for these indicators come from a variety of considerations on basic data sources. However, it is challenging to define algorithms for source evaluation based on data resulting from services aggregating and orchestrating other sources [25, 22]. Ascertaining provenance means ensuring that the source of data is verifiable, i.e., that it corresponds to the one declared in the process of creation. Trustworthiness is intended as the possibility to ascertain the correctness of the information provided by a data source, which is loosely related to provenance [16]. Ideally, but infrequently, data samples can be independently measured by different users, thus allowing cross-checking and error correction. For original data, i.e., provided by its creator, the trustworthiness score is usually derived from the reputation of the creator. Clearly, guaranteeing data quality, provenance and trustworthiness is not enough, it is necessary to ensure that the computation is correct and that no useful results are hidden (completeness).

3.1.3. User

The last category of services of tier I is not specific to mobility but it contains essential functionalities for the other two categories. The most representative case is that of User Profiling and Management. User profiling is not required to create services for mobility, but it has become essential to ensure usability, to provide user assistance, and to even anticipate and plan for the next movements of the user (cf. Google Now[3]).

Data theft. Here, the most obvious threat regards the possibility of stealing information derived from the profile dataset, such as preferences, recordings of movements, orders and payments.

Countermeasures. In our setting, a possible approach is to empower the user with control over its profile and the related access policies [3].

3.2. MaaS Stack | Tier II

3.2.1. Business Intelligence

The second tier of the MaaS Stack adds a new category next to the ones of the first tier: Business Intelligence, i.e., services exclusively dedicated to provide insight on the usage and performances of services of the first tier.

This services can implement any kind of data mining algorithm useful for monitoring the profitability, sustainability, and reliability of the provided services, as well as for determining trends and making predictions on future usage, for capacity planning and policy definition. Most of this algorithm and services do not usually works on the physical devices but they are part of a Cloud Architecture.

[3] https://www.google.com/search/about/learn-more/now/

Business Intelligence Data Theft. Business Intelligence analyses are important source of sensitive information for insiders (also in this case Managers with privileged access) that could expose relevant data to third parties. Indeed, without Business Intelligence services it would be very difficult or even impossible for insiders to obtain such data, that otherwise would require the access to massive amounts of private information over long periods.

Managers of Business Intelligence services can apply targeted analyses to infer reserved information, such as policies and business strategies of their company. An example of this type of attack is what we simulated in [10], where by just analyzing the database of validated tickets of a public transport company of the urban area of Bologna, we were able to reconstruct the distribution of the various types of tickets in the different zones of the city.

Countermeasures. SMAll serves the purpose of mediating the access to relevant data for Business Intelligence. Every operator wishing to obtain statistics or performance indicators about its own services can freely create instances of the platform-approved analytics services.

Regarding mitigation, the most effective way to hinder the possibility to misuse Business Intelligence services is to properly sanitize the datasets and to control the workflow of this information. These techniques [48] aim to prevent insiders from correlating Business Intelligence services with external data sources to derive hidden patterns or de-anonymize sensitive information.

3.3. MaaS Stack | Tier III

The third tier of the MaaS Stack is that of MaaS operators, i.e., eMobility operators that use services of other companies, traded within a federated market. In our case, SMAll gives support to such a market but the creation of dynamic federations of MaaS operators rises specific threats within SMAll (and MaaS markets in general).

In this scenario the main issues to consider are:

- Data service management to avoid manipulation, impersonation, and sensitive pattern discovery (Prevention and Detection);
- Service workflow management to monitor invocation trends of services (Mitigation and Detection);
- Service quality and trustability management to verify the correctness of the service results (Prevention and Detection).

Indeed, the PaaS layer in SMAll differs from most PaaS solutions. Traditionally PaaS provides offer execution environments that isolate tenants. On the contrary, SMAll is built to ease the publication, integration, and orchestration of services owned by different operators.

A simple example to clarify this characteristic is a one-stop ticketing application that orchestrates:

- a dynamic planner service providing routing options;

- a user profile manager to sort them according to user preferences;
- a real-time availability seat reservation service;
- a set of services for payment.

The hierarchy of the ticketing service spans many layers, e.g., it integrates the dynamic planner that, in turns, orchestrates many services for static (mapping, timetables) and real-time data (delays, planned extraordinary events, disruptions). The composition of services forms a tree of dependencies that reaches the level of raw-data information services, possibly branching within the domains of different companies.

Since SMAll aims at supporting this kind of interoperability, we argue that it shall also assume responsibility for the trustworthiness and reliability of the services; this is unusual for traditional PaaS [43]. Moreover, access control policies can be heterogeneous, exchanged data can have different sensitivity levels, and the agents can be competing operators.

Clearly, the main insider threat for this scenario comes from the service providers themselves, the MaaS operators. The malicious goals can be of various kinds, spanning from the de-legitimization of services of competing operators, to the theft of stored information such as policies or business strategies, to insiders that apply mining techniques to infer these information using the data available from their vantage point.

We now proceed by focusing our analysis on the relevant insider threats within the categories of the third tier of the MaaS Stack.

3.3.1. Roaming and Clearing

SMAll aims at providing interoperability between different operators. In this context, interoperability means that it is possible to implement ticketing systems which seamlessly work on different operators across their zones of influence. As mentioned in § 2, this concept (and the category of services that supports it) takes the name of Roaming. Usually, to support at a business level the roaming for users among operators, business agreements should be put into place to implement a Clearing system for the redistribution of profits between transport operators. In this Section, we consider threats as directed to the Clearing category since it comprises also the threats to the Roaming one.

Pattern Extraction. As analyzed in [10], the need for Clearing services is satisfied through a centralized (federation-wise) system able to collect all the different data sources from different operators and to perform economic evaluations. A centralized clearing system scenario is typically based on a central database that collects all the ticket validation data from every public transport operator. This database is used both to perform economic evaluations to redistribute profits and to store a permanent proof of the validity of this evaluation. The clearing system must fulfill an effective trade-off between public verifiability of the correctness of its operation and protection of sensitive data provided by operators. As the last cited work shows, an insider can perform data mining analysis and pattern discovery on the tickets datasets in order to retrieve sensitive information about business strategies and perform unfair competition.

Countermeasures. To counteract *Pattern Extraction*, it is possible to deploy sanitization techniques [49] able to mask the data enough to deny the possibility to perform pattern analysis. These sanitization techniques balance the needs of masking sensitive data and keeping enough properties and information to perform the economic evaluations. In order to do what we described, we could assemble an anonymization system, that combines masking techniques for the raw dataset (once deployed in the centralized database clearing system) and a differential privacy engine able to introduce a certain amount of noise and prevent exploit techniques as cross-combining data with external ones.

3.3.2. Access Control and Service Level Agreement

Service Level Agreement (SLA) and Access Control (AC) services in SMAll are meant to throttle the invocation of tier I services provided by an operator on the basis of commercial agreements with other operators. It is possible to see SLA as a contract ruling the quantity or rate of invocation of each service, and AC as a contract ruling the quality or the set of provided data or services. Obviously, malicious insiders may try to circumvent these limitations.

Countermeasures. When a SLA or an AC policy is in place, all service invocations must be tracked (or even proxied) by an infrastructural service provided by SMAll. This makes evading enforcement difficult. The most common vulnerability in this context is not tied to policy enforcement, however, but rather to policy specification. To this end, SMAll could restrict acceptable policies to those drafted with an internal helper service, following a standard framework, and formally verifying their soundness before applying them. Access control models and formal policy specification languages have been around for some time [62, 18], and they have evolved into sophisticated, standardized models like ABAC [39, 61]. Inadequate (but consistent) policy definitions due to poor understanding of the federation interactions or to carelessness cannot be tackled at this level; logging and auditing facilities integrated in SMAll provide valuable feedback at run-time about the effectiveness of installed policies.

3.3.3. Business Intelligence

Similarly to tier II, in tier III we have a category of services dedicated to business intelligence. The difference with respect to the services of the second tier is that here the analyses span data belonging to a multitude of operators. Indeed, as it happens for clearing services, the business intelligence services of the third tier relate to the management of data, statistics, and administration of services shared among operators. The availability of such aggregated data can give free access to companies (seen as federated insiders) to data

and analyses of competitors. Referring again the case of the dynamic route planner as a running example, the service can use real-time data of different companies to take into account the average delays of transport vehicles in the calculation of its solutions. The averaged delays are the result of a business intelligence service that collects all the delays of a route within a specific area that involves several operators and calculates the delays. Finally, the recorded delays are collected into a shared dataset accessible by all the participants.

In this example, an insider can use the collected dataset to find out where the competitors operate with bigger delays and profit from this information by exposing their faults to the regional administration. Insiders can also expose cartels where operators systematically provide a bad service during rush hours to favor a specific company (e.g., because they hold some economic interest in it). Finally, the insiders can also find out if an operator hides delays making analysis on the correspondent road conditions (e.g., showing that buses could not sustain certain speeds since their routes were jammed).

Countermeasures. All the countermeasures for this kind of attacks are based on a trade-off between the amount of sensitive data preserved and utility of the queries. Different anonymization and sanitization techniques have been proposed for complex datasets, but since in SMAll Business Intelligence services share the results of queries, we need to introduce a measure that indicates the maximum amount of anonymized information such that the queries still work.

Different works proposed metrics for the evaluation of the amount of privacy preserved in specific dataset. A measure introduced in [54] defined an evaluation metric about the presence of pattern in a dataset called δ-presence. We can use this metric to evaluate the presence of a specific patterns in the shared dataset. Another interesting work in this direction is [36] which operates by complementing existing techniques with post randomization methods.

4. CoTs and MaaS: Insider Threats and Solutions

After our general analysis of insider threats in MaaS, we concentrate on Cloud of Things (CoT): one of the main enabling technologies for MaaS. We mentioned the role of CoTs in MaaS and the threats linked to their ubiquity and continuous connectivity in § 1–3. Recognizing that a solution at the low level of CoTs could positively impact the security of MaaS services, we deepen here our analysis on insider threats of CoTs, in the context of MaaS, and propose an overlay architecture over networks of CoTs to mitigate some of the identified threats.

In general, the Internet of Things [4, 33] and the system of networks of IoT devices that constitutes the, so called, *Cloud of Things* [5, 38], relies on the idea of a world-wide network of interconnected entities. Concretely, these entities are heterogeneous elements interacting over disparate systems of networks: the definition extends to comprise human beings and computers as well as general-purpose environmental sensors (light, humidity, temperature, sound) to specific devices like road traffic monitors or GPS trackers. Making all these entities interact with each other had and is having sensible, successful applications in multiple domains like automotive, health-care, logistics, environmental monitoring, and many others.

CoTs play an important role in enabling many of the modern features of MaaS services, indeed, entities within the context of CoT share three common denominators [59]. They are:

- *locatable* at multiple layers, spanning from their position within an interconnected network to their actual geographical location;
- *addressable* in such a way that they accept connections from other entities;
- *readable* other entities can query them to obtain some information.

If on the one hand these properties constitute the promising characteristic of CoT, on the other hand they make CoT-based networks open to many kinds of malicious attacks conducted by a plethora of possible adversaries.

4.1. Cloud of Things and MaaS: Insider Threats

Considering the categories of threats analyzed in § 3, we summarize a brief account of the possible attacks in the context of CoT linked to insider activities:

- *Data Leakage* entities can accidentally release private or sensitive data to unauthorized entities. Malevolent attackers could also gain information from "alternative" usage of sensors, e.g., by employing temperature, light or audio sensors to check the presence of people;
- *Crowdsourcing Attacks and Data Manipulation* where entities that publish information feed fabricated data to change the behavior of the services that rely on them;
- *Device Misbehavior* insiders can exploit weaknesses in the protocol of interaction of entities, e.g., sensors and actuators, to cause malfunctions and hardware failure.

Given the threats above, we investigated whether the networking architecture of CoT entities and its working protocols could mitigate and counteract some of the vulnerabilities analyzed in Section 3.

The idea has already been speculated in literature, for example, in [59] the authors dissect the issue as driven by the two principles of *location of intelligence* and *degree of collaboration*. The first indicates where the intelligence resides in the network, i.e., if edges of the network provide services rather than simple data. The second regards the degree of interconnections among heterogeneous entities, i.e., if they are

mainly partitioned based on their nature (sensors for probing, servers for collection and manipulation of data, etc.) or if they interact on a peer-to-peer basis. Combined, these two factors characterize the architecture of the network interconnecting the considered entities. Mainstream solutions follow a centralizing approach, leaning towards "simple" entities that mainly collect information at the edges of the network and centralized, possibly hierarchical, hubs that aggregate, manipulate, and redistribute available data.

At the other end of the spectrum, there are decentralized systems that leave some freedom at the edges of the network, allowing entities to take some decisions. Entities can also constitute partitions with some emerging intelligence (e.g., regarding probing times, correction of sensing, etc.) without a direct control and sharing no information with a central, high-level entity.

From an insider threat standpoint, we argue that the centralizing approach constitutes the most dangerous configuration. Indeed, following such an approach, entities at the edge of the network are passive elements. They are open to any kind of query. Malicious ones could be used to infer private information and even cause malfunctions on the devices, e.g., by consuming their batteries or causing hardware failures due to overuse. Moreover, the approach is prone to the well-known issue called "central point of failure" where users of the network must connect to the central hubs, which in turn could be compromised to cause denials of service as well as to reveal sensible information of all users.

On such observation, we investigated networks characterized by a weakened degree of centralization and an increased collaboration among the entities. In such architectures, entities at the edges can process local information and provide it to both central hubs as well as to other peers. These decentralized networks enjoy a stronger degree of security as nodes can refuse to collaborate in requests that are deemed dangerous or deviate from established, secure protocols of interaction. In addition, if proper dynamic reconfiguration techniques are applied, even in case of malfunctioning central hubs, the local services remain accessible.

To concretize the observations here discussed, we proceed to present a distributed networking architecture over CoTs, tailored to the case of MaaS. In the proposed architecture, entities collaborate on a local scale, while hubs (gateways and administrators of geographical zones) mainly work as routing nodes to interconnect the edges. The architecture assumes the form of an overlay network, where gossip-based algorithms regulate the diffusion of information over the network.

4.2. An Overlay Network Approach

As previously highlighted, data on mobility, if not properly managed and/or anonymized, can become an important vehicle for privacy leak attacks. In the context of CoT, we also underline that among the wealth of data users yield to their platform of choice, those related to the handling of real-time information are typically the most sensitive ones and the most difficult to protect. On the one hand, real-time information requests, generated by users, help building a precise correlation between their position, time, and current activity. On the other hand, it is difficult to obfuscate the details enough to make these correlations unfeasible, without also lowering the utility level of the results.

In our investigation, we also considered how our proposed platform for a market of mobility services (SMAll) would impact on the choices made by developers for CoT networks. Indeed, although in our discussion on SMAll we did not provide a specific model to structure services for the connection and collection of data, we highlight that the main developing model for such systems is that of orchestration [21], which tends to centralize the data and their elaboration to the central nodes of the systems. We discussed the shortcomings of such hierarchical approach in Section 3.

To proceed our exploration, we deem useful to recall a macroscopic classification of mobility services into two categories.

- *Information services* have the responsibility to provide general-purpose information aseptically and as provided e.g., by devices installed by transport operators or user personal devices. These information are not usually customizable, meaning that they are not provided in a tailored fashion based on the user request, such as the current position of buses, timetables, availability of parking spots, etc.;
- *Data processing services* compute data from different sources, combining what is provided by information services and personal information given by users. They produce an answer in form of processed data, tailored to a specific user need, e.g., a travel path, the presence of points of interest along the path that match the user's preferences or the suitability of each path section to the user's special needs, if any.

In our proposition of an architecture for CoT, we strive also to empower entities with control over the data flowing from the category of information services, so that the structure of requests is hidden from any central authority, relying instead on the particular construction of the network, where neighboring nodes can satisfy the request of nearby (wrt to the overlay position) entities. Our approach breaks this direct link between data storage and request elaboration.

4.3. Gossip-based networks

Before describing the details of our proposed architecture, we provide a brief account on gossip-based networks, whose principles are at the basis of our overlay network, described in the next section.

Gossip communication [34, 7] is a style of computer-to-computer communication protocol inspired by the form of gossip seen in social networks. Modern distributed systems often use gossip protocols to solve problems that might be difficult to solve in other ways, e.g., either because the underlying network has an inconvenient structure or is extremely

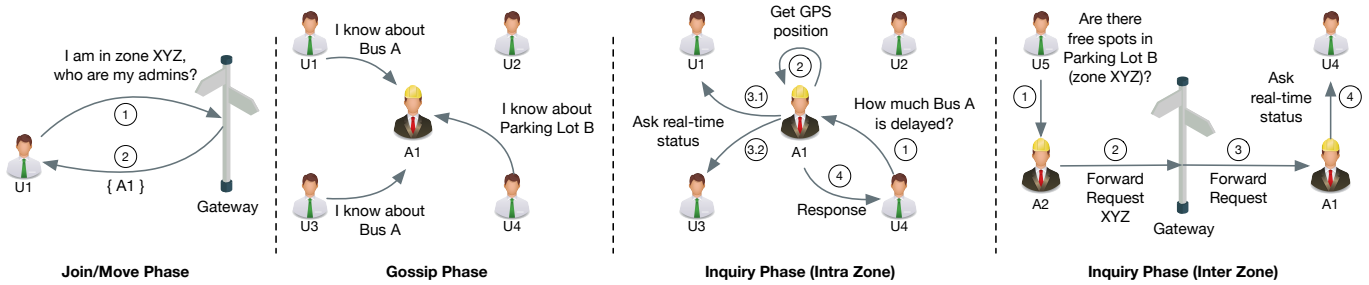

Figure 4. Phases of the Gossip Network.

large. Computer systems typically implement this type of protocol with a form of random "peer selection": with a given frequency, each machine picks another machine at random and shares any hot rumors.

The power of gossip lies in the robust spread of information. Even if a node had trouble understanding another node, the former will probably run into someone else soon and can learn the news that way. Expressing these ideas in more technical terms, a gossip protocol is one that satisfies the following conditions:

- the core of the protocol involves periodic, pairwise, interactions;
- the information exchanged during these interactions is of bounded size;
- when agents interact, the state of at least one agent changes to reflect the state of the other;
- reliable communication is not assumed;
- the frequency of the interactions is low compared to typical message latencies so that the protocol costs are negligible;
- there is some form of randomness in the peer selection. Peers might be selected from the full set of nodes or from a smaller set of neighbors;
- due to the replication, there is an implicit redundancy of the delivered information.

On these principles, we can find three prevailing styles of gossip protocols:

- *Dissemination protocols (or rumor-mongering protocols)* which use gossip to spread information; they basically work by flooding agents in the network, but in a manner that produces bounded worst-case loads. Event dissemination protocols use gossip to carry out multicasts. They report events, but the gossip occurs periodically and events don't actually trigger the gossip. One concern here is the potentially high latency from when the event occurs until it is delivered.
- *Background data dissemination protocols* continuously gossip about information associated with the participating nodes. Typically, propagation latency is not a concern, mainly because the information in question changes slowly or there is no significant penalty for acting upon slightly stale data.
- *Anti-entropy protocols* repair replicated data by comparing replicas and reconciling differences. These protocols compute a network-wide aggregate by sampling information at the nodes in the network and by combining the values to converge to a system-wide value — the largest value of some measurement, the smallest, etc.. The key requirement is that the aggregate must be computable by fixed-size pairwise information exchanges; these typically terminate after a number of rounds of information exchange logarithmic in the system size. As a side effect of aggregation, it is possible to solve other kinds of problems using gossip; for example, there are gossip protocols that can arrange the nodes in a gossip overlay into a list sorted by node-id (or some other attribute) in logarithmic time using aggregation-style exchanges of information. Similarly, there are gossip algorithms that arrange nodes into a tree and compute aggregates such as "sum" or "count" by gossiping in a pattern biased to match the tree structure.

Many protocols that predate the earliest use of the term "gossip" fall within this rather inclusive definition. For example, Internet routing protocols often use gossip-like information exchanges. A gossip substrate can be used to implement a standard routed network: nodes "gossip" about traditional point-to-point messages, effectively pushing traffic through the gossip layer. Bandwidth permitting, this implies that a gossip system can potentially support any classic protocol or implement any classical distributed service. However, such a broadly inclusive interpretation is rarely intended. More typically, gossip protocols are those that specifically run in a regular, periodic, relatively lazy, symmetric, and decentralized manner; the high degree of symmetry among nodes is particularly characteristic. Thus, while one could run a 2-phase commit protocol over a gossip substrate, doing so would be at odds with the spirit, if not the wording, of the definition.

In our investigation, we found that none of the protocols described above constitutes a comprehensive solution. Hence, we followed an hybrid approach on the definition of the architecture of our proposal. In our overlay network nodes disseminate not the actual knowledge (information) but the possibility to retrieve it — e.g., a user is on a bus and notifies nearby peers that there exists a node that can provide information of the position of the bus in object. On the other hand, moving nodes periodically probe the network to join the partition to which they geographically belong, this is done following a background dissemination approach. Finally, when multiple nodes can respond to a request of a peer, the data they provide is passed in an aggregated, weighted form, considering the trustability of the node that emitted the datum. This recalls anti-entropy approaches that focus on providing a system-wide consistent observation as aggregate of many local responses.

The concepts above are exemplified in Figure 4. From left to right, when users want to join for the first time or move between zones, they query known gateways to obtain the address of the administrator of the geographical region to which they currently belong. Administrators act as local authorities to connect users within the same zone and, since they are meant to be implemented by municipalities, they can also provide their trustworthy data to user requests. In the gossip phase, users disseminate to each other (within the same zone) and to administrators information regarding what requests they are willing to answer. In Figure 4, U1 and U3 declare to know something about the Bus A (e.g., its position, delay), while U4 has some information on Parking Lot B (free spots, occupancy). The remaining phases regard the inquiry of available data. The two phases are distinguished by whether they cross the boundaries of the same zone. If requests are bounded within the same zone, the administrator forwards the requests to the available users, it aggregates their responses with its own data (if available) and sends the response back to the invoker. In the second scenario, requests are forwarded among administrators and gateways to reach the requested zone and look for available peers to answer them.

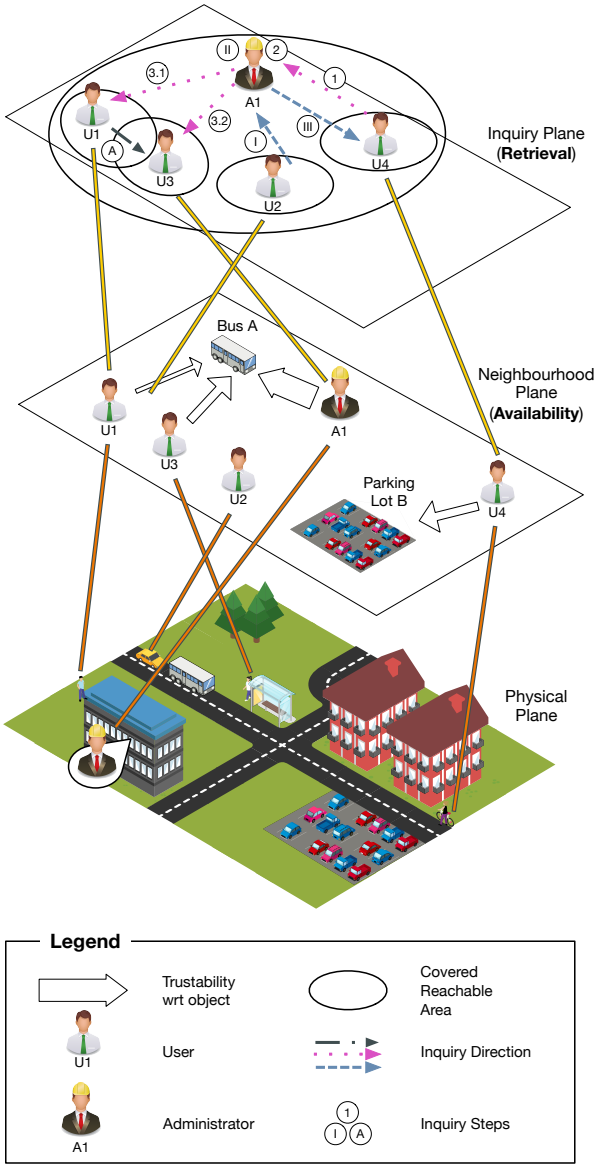

Figure 5. Depiction of the Overlay Gossip Network.

4.4. A CoTs overlay network for MaaS

The overlay network that we propose in this work requires a software component: an application installed on the device that allows users to start the process of dissemination of information within a peer-to-peer gossip-like network. Here, as in peer-to-peer networks, each entity can be both a connecting node or an endpoint node that owns the final information. The overlay is then composed as a hierarchy of nodes collecting the dissemination of requests from the output nodes. Such collection is temporary and regards just the possibility to answer a question regarding the physical world. This means that the interims and locals are aware of what type of information a node contains, however they do not know the actual data.

Figure 5 shows the two planes, over the physical one, that characterize the proposed overlay network. From the physical plane, users join the top-most inquiry plane where they notify other nodes of their willingness to answer. In-between the two planes, the Neighborhood plane represents the knowledge of the users and they trustability (the thicker the arrow, the more trustable the peer).

Below we describe two use cases, where we show how our solution improves the security of a mobility service that displays information in real time.

Bus delay. A service that exposes the real-time delays is typically based on an algorithm that predicts the estimated delay based on the GPS position of the vehicle. This information is then saved on a centralized storage where the delay is calculated. As described above, however, this methodology introduces security problems on the storage of data and quality of service, entailed by relying exclusively on the bus GPS loca-

tion, which possibly could be inaccurate. With the proposed approach, we show how it is possible to improve the safety and performance of this type of service.

In Figure 5, U4 (Inquiry Plane) wants to know the delay of Bus A (1). It does not have nearby nodes, so it forwards the request to A1. A1 has information regarding Bus A, provided by the system discussed above (2). Since U1 and U3 are within the reach of A1 and they previously gossiped that they can contribute to answer the request (arrows on the Neighborhood plane), A1 queries them ((3.1) and (3.2)), it collects their data, it aggregates them, and it responds to U4.

As shown in Figure 4, when the user jumps on the bus, it notifies its status and its willingness to answer questions regarding its surroundings.

Although users U1 and U3 remain anonymous, the requesting user U4 can rate the quality of their answer, a metric that can feed the system that assigns the degree of trustability to users. Since such issue is orthogonal wrt the specifics of our network, we leave it as a future work. Besides security, the system enjoys a finer grade of precision on the reporting of the information as the machine-generated data from A1 is integrated with user-generated, close-to-the-source (Bus A, in the example) information.

Parking lot availability. Another use case, where our approach improves security and precision, regards the research of parking spots. Generally the only services that offer to check real-time availability of parkings are provided by specific hardware to be installed on each spot. Most parking areas do not have this type of hardware but such a real-time service could sensibly alleviate traffic, also saving fuel, and the time of the drivers.

A solution, relying on a peer-to-peer decentralized network where nearby users can provide information on the status of parking areas, is a desirable one.

In Figure 5, this case is represented by user U2, which is driving a car. In the Inquiry plane, U2 asks to A1 if it knows whether there are free spots in Parking Lot B (I). Although A1 does not have such information, U4 notified that it could answer such request (II). In this case A1 collects the data from U4 (III) and forwards it back to U2.

Although in this case only U2 responds to the invocation of U4, the response does not include data that can directly identify U2. However, we recognize that when the number of users that provide aggregated information is small, they become easily recognizable. For example, in the case presented here, U2 could rate the information provided by U4 in such a way that, although without disclosing the identity in the network of U4, its associated rating (trustability) could become a marker to identify it. In these regards, we propose to tackle the issue in the future, by investigating rating systems that, with proper entropy and delays, allow to scramble the marking of users.

4.5. Analysis of Mitigations and Future Steps

After having presented our solution for MaaS, based on an overlay network of CoTs, we dedicate this section to a thorough analysis on the mitigations put in place by the proposed architecture. In particular, we refer to the threats analyzed in Section 3 and indicate the effectiveness of the overlay network in mitigating them. This is further illustrated in Figure 6, which updates the table in Figure 3 with a column "Mitigation Level", reporting the impact that the overlay network has on the identified threats.

A safe level of decentralization of computing and data storage can constitute a sensible improvement with respect to many security problems [56]. The first aspect to note regards data management. If with a centralized system all sensitive data, such as present and past location and personal identifiers, are stored in a single place, with our solution these pieces of information are stored just in the intermediate nodes and only in the form of metadata about which nodes hold a certain type of information . The actual contents are exchanged only in an anonymized, possibly aggregated form, in the inquiry phase of the interaction.

There are many advantages with this kind of approach. Considering table 6 we can see as there are different "levels" (represented with 3 different colored icons) of mitigation introduced with the proposed overlay architecture.

Green level, check mark (✔) icon: all the Threats and the Agents that, thanks to the overlay network proposed, we can strongly mitigate. Most of them are threats and kinds of events based on the exploitation of data management vulnerabilities. All attacks resulting from the deep analysis of entire data datasets as Extraction Patterns; Data extraction; Data exploitation through crossing of data 3.2.1 are strongly limited. This is a direct consequence of not storing data in a centralized database. Data is generated at a specific moment in the form of a targeted request by a user, and the only information that is temporarily logged is the metadata for who created that particular data, but not the corresponding data. An output node could still gather data in this scenario, recording all the data that came after a request on the network; however, both targeted attacks and wide collections aimed at correlating personal data are made less effective, since the connection between the real information and the metadata of the output node is created at the time of the request by the intermediate node, and it is not predictable. Data Mining algorithm and pattern extraction as well as every supervised machine learning algorithm need a training set derived from a huge real dataset: being able to reduce the dataset surface, we are able to greatly reduce the feasibility of these attacks.
For similar reasons, in this scenario theft and leakage 3.1.1 of data are threats with significantly reduced impact, because less sensitive data are stored, to begin with, and when they are stored they include only metadata and not the real information. Data alteration and manipulation as referred to in 3.3 is also a strongly limited threat. In this case, the main mitigating factor at this level of MaaS derives from the increased difficulty of manipulating the origin of a given piece of data. If, in the original architecture, an insider was able to forge the origin of a data element, thus spreading false but

Tier	Agent	Agent Type	Insider Threat	Mitigation Level	Event Type
1 & 2	User	Competitor, Untrained/Distracted Insider, Outward Sympathizer	Fake Data Injection	✖	Sabotage
			Service Behavior Manipulation	!	Product Alteration, Sabotage
	Unauthorized Guests	Competitor, Theft, Activist	Man-in-the-middle Attack	!	Misuse
	Developers	Competitor, Partner, Disgruntled Insider	User Impersonation	!	Sabotage, Espionage, Misuse
	Service Administrators	Partner, Disgruntled Insider, Untrained/Distracted Insider, Supplier	Insider Impersonation	!	Sabotage, Espionage, Misuse
	Service Managers	Partner, Disgruntled Insider, Untrained/Distracted Insider, Supplier	Crowdsourcing Attacks	✖	Sabotage, Financial Fraud
	Agent after privilege escalation	Activist, Competitor	Data Leakage - Accidental - Data Theft - Resale of Data and Access - Business Intelligence Data Theft	✔	IP Theft, Opportunistic Data Theft, Physical Theft, Accidental Leak
3	Federated MaaS Member	Competitors Nation State Partner Supplier	Data Manipulation, Trustability, Tampering of Data Provenance, Data Trustworthiness	✔	Financial Fraud, Product Alteration
			Service Behavior Manipulation	!	Financial Fraud, Product Alteration
			Composition of Unverified Services and Data	!	Misuse, Sabotage, Espionage, Product Alteration
			Denial of Service	✖	Sabotage
			Service Workflow Manipulation	!	Misuse, Sabotage, Espionage, Product Alteration
	MaaS Competitor	Nation State, Partner, Supplier	Data Analysis: - Pattern Extraction - Data Mining - Data Exploitation trough data crossing	✔	Accidental Leak, Opportunistic Data Theft Espionage, Financial Fraud
	Helper Service	Competitors, Nation State, Partner, Supplier		✔	

Figure 6. Update of the summary table in Figure 3. Column "Mitigation Level" (second from the right) reports the impact of mitigations in the overlay network wrt pre-existing threats.

credible information, in the overlay network, every data is created at the time of the request, when the source was already certified on the server of presence. Two aspects make this scenario better than the original one:

1. It is not possible to precisely manipulate the origin of a data. If before an attack between a competitor in a MaaS architecture consisted of falsifying entire targeted services acting at binding time, now it is much more difficult to determine precisely when and where a specific service will be required.
2. Although it is partially possible to manipulate the source of specific data, the whole "p2p philosophy" is to never consider any source totally reliable, building trust in a piece of information through cross-checks between multiple requests, either to verify it or to mediate it with other results, thus lowering the impact of malicious elements.

However, it would not be right to say that a tier 3 insider can no longer manipulate a service by falsifying the workflow, manipulating its behavior, or composing it with unverified services. These attacks are still possible because the overlay network does not act at such a high level of service certification, the federation of services remains bound to policies that the two federal operators agree. However, the new scenario reduces potential impact of attacks rather than reducing the vulnerability to an attack.

Yellow level, exclamation mark (!) icon represents these types of insider threat which enjoy this kind of weaker mitigation than the previous category, but still achieve important positive effects and show potential for future improvements.

The considered attacks include all the service structure attacks as Service Behavior Manipulation, Service Workflow Manipulation or result manipulation where the insider interacts by attacking the entire exposed service or by creating one that is purposefully vulnerable. With the overlay network scenario, these attacks are still possible since there is no longer a form of formal certification and control over the composition of federated services as in the previous case. In fact, the overlay network acts only at the level of data sharing and processing, and this allows to check for a possible anomaly in a timely manner. Future development arising from this problem might be to implement a formal verification mechanism of a service, which describes workflow and acceptable behaviors for each particular service category and verifies the reliability of new ones.

Another category of insider attacks mitigated is the one related to the possibility to act as another user and perform impersonation. Whether it is user impersonation or insider impersonation linked to a MITM attack, these are effectively limited by our architecture. The two fundamental safeguards are:

1. Acceptance to the network as a gossip protocol network node can and must be governed by precise access policies. The peculiarity is that, unlike an open service or cloud, it is easier to implement these access policies on a peer-to-peer-like network than the one it is proposing.
2. Impersonation attacks and MITM are targeted attacks. Since the creation of data is linked to the exact time of creation, it is much more difficult for an attacker to foresee the exact moment for intercepting the request; only after the request has been made, the attacker will be able to determine the nature of data and create it coherently.

Red level, cross mark (✖) icon highlights those threats that, already present in the standard centralized scenario, remain substantially unchanged in the proposed architecture, or are only slightly mitigated.

The first case is denial of service. On the one hand, in this kind of peer-to-peer network the ability of single nodes to resist a DDoS attack is reduced by (typically) limited bandwidth and processing power. On the other hand, considering

the fact that we do not have a single node for every service, there is no single centralized storage so we avoid to expose the system to a single point of failure attack. This mean that the denial of service of a single node does not necessarily imply the end of the full service. In absence of a formal analysis of the system's behavior, we refrain from claiming that our architecture mitigates DDoS attacks, leaving deeper studies to future works.

Falsification of data in crowdsourcing deserves a different discourse. Even this type of attack is not mitigated by the overlay network, since crowdsourcing and crowdsensing data are automatically acquired once the node is properly logged on the network, and not personally by the user. For this reason, the attack remains the same if the crowdsourcing engine of each single node is manipulated. It should be noted, however, that it is worth studying how to extend the reporting and feedback system even at a the real-time level by verifying the reliability of the data in the post-creation phase.

If instead we consider the performance of the service quality, we found considerable improvements in the accuracy of the result and in the amount of computation required. The case of the bus delay is significant. Previously, these data could have been the result of some algorithm that calculated an estimate, based on the position of the bus and possibly some historical data on previous delay. Such estimates could be too conservative, skewed, or otherwise biased by whatever behavior the operator adopts, including scarce commitment to the service. On the contrary, integrating data from users close to the source of information and willing to contribute can make the estimate more precise.

Despite this, however, we do not expect a better overall performance, that is, in terms of efficiency since computing the added security levels make the demand/response certainly more laborious and slow. Despite this, given the importance and sensitivity of the data processed, a worsening of efficiency (but not quality) of these categories of services can be offset by a greater security.

The use case of the parking availability goes on the same direction, and has already been successfully adopted in smartphone applications [60].

5. Conclusions

In this paper, we revised and extended previous work where we presented the concept of Mobility as a Service and how MaaS operators shall facilitate the dynamic provisioning of multi-modal transportation to their users. To support such flexibility we are developing a federated marketplace of services called SMAll, aimed at harmonizing data flows and service invocations.

This kind of federated platform is particularly sensitive to insider threats, which emerge at different layers, targeting both the constituent components provided by users and operators and the services provided by the platform itself.

The MaaS Stack, our tiered view on the components of MaaS markets, allowed us to treat in isolation the security issues of each tier. Often, these issues turn out to be instances of well-known threats in the fields of cloud computing, service-oriented architectures, supply chain management, and trusted business partnerships.

In principle, the platform allows to implement context-specific versions of the solutions proposed in the literature regarding the aforementioned fields, as well as novel solutions inspired by their cross-fertilization. We argue that the central role of SMAll in mediating every interaction and in collecting their traces makes the platform fit to host solutions to the presented security issues of MaaS markets.

In addition to our general treatment on insider threats related to the context of MaaS markets, we also considered the perspective of the Cloud of Things, presenting an architecture that constraints the quality and quantity of data that an insider could obtain from users, also optimizing the routing of requests to only those users able to answer them.

References

[1] Qutaibah Althebyan. *Design and analysis of knowledge-base centric insider threat models*. ProQuest, 2008.

[2] Qutaibah Althebyan, Rami Mohawesh, Qussai Yaseen, and Yaser Jararweh. Mitigating insider threats in a cloud using a knowledgebase approach while maintaining data availability. In *ICITST*, pages 226–231. IEEE, 2015.

[3] Kai Kuikkaniemi Antti Poikola and Harri Honko. Mydata a nordic model for human-centered personal data management and processing. Technical report, Ministry of Transport Finland, 2014.

[4] Kevin Ashton. That 'internet of things' thing. *RFiD Journal*, 22(7): 97–114, 2009.

[5] Debasis Bandyopadhyay and Jaydip Sen. Internet of things: Applications and challenges in technology and standardization. *Wireless Personal Communications*, 58(1):49–69, 2011.

[6] Matt Bishop. Position: Insider is relative. In *Proceedings of Workshop on New security paradigms*, pages 77–78. ACM, 2005.

[7] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE/ACM Trans. Netw.*, 14(SI):2508–2530, June 2006. ISSN 1063-6692.

[8] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *FGS*, 25(6):599 – 616, 2009. ISSN 0167-739X. doi: http://dx.doi.org/10.1016/j.future.2008.12.001. URL http://www.sciencedirect.com/science/article/pii/S0167739X08001957.

[9] Rajkumar Buyya, Rajiv Ranjan, and Rodrigo N Calheiros. Intercloud: Utility-oriented federation of cloud computing environments for scaling of application services. In *AAPP*, pages 13–31. Springer, 2010.

[10] Franco Callegati, Aldo Campi, Andrea Melis, Marco Prandini, and Bendert Zevenbergen. Privacy-preserving design of data processing systems in the public transport context. *Pacific Asia Journal of the Association for Information Systems*, 7(4), 2015.

[11] Franco Callegati, Saverio Giallorenzo, Andrea Melis, and Marco Prandini. Insider threats in emerging mobility-as-a-service scenarios. In *50th Hawaii International Conference on System Sciences, HICSS 2017, Koloa, HI, USA, January 4-7, 2017*. IEEE Computer Society, 2017.

[12] Sabrina Capitani Di Vimercati, Sara Foresti, and Pierangela Samarati. Data protection in cloud scenarios. In *Revised Selected Papers of the 10th International Workshop on Data Privacy Management, and Security Assurance-Volume 9481*, pages 3–10. Springer-Verlag New York, Inc., 2015.

[13] Tim Casey. A field guide to insider threat. Technical report, Intel, 2015.

[14] Youngho Cho, Gang Qu, and Yuanming Wu. Insider threats against trust mechanism with watchdog and defending approaches in wireless sensor networks. In *SPW*, pages 134–141. IEEE, 2012.

[15] William R Claycomb and Alex Nicoll. Insider threats to cloud computing: Directions for new research challenges. In *ACSAC*, pages 387–394. IEEE, 2012.

[16] Chenyun Dai, Dan Lin, Elisa Bertino, and Murat Kantarcioglu. *SDM*, chapter An Approach to Evaluate Data Trustworthiness Based on Data Provenance, pages 82–98. Springer, Berlin, Heidelberg, 2008. ISBN 978-3-540-85259-9. doi: 10.1007/978-3-540-85259-9_6. URL http://dx.doi.org/10.1007/978-3-540-85259-9_6.

[17] Mila Dalla Preda, Saverio Giallorenzo, Ivan Lanese, Jacopo Mauro, and Maurizio Gabbrielli. AIOCJ: A choreographic framework for safe adaptive distributed applications. In *SLE*, pages 161–170. Springer, 2014.

[18] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. In *IWPDSN*, POLICY '01, pages 18–38, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-41610-2. URL http://dl.acm.org/citation.cfm?id=646962.712108.

[19] Sabrina De Capitani di Vimercati, Sara Foresti, and Pierangela Samarati. Data security issues in cloud scenarios. In *International Conference on Information Systems Security*, pages 3–10. Springer International Publishing, 2015.

[20] Gary Doss and Guvirender Tejay. Developing insider attack detection model: a grounded approach. In *ISI*, pages 107–112. IEEE, 2009.

[21] Nicola Dragoni et al. Microservices: yesterday, today, and tomorrow. In *Present And Ulterior Software Engineering*. Springer, 2016. to appear.

[22] Schahram Dustdar, Reinhard Pichler, Vadim Savenkov, and Hong-Linh Truong. Quality-aware service-oriented data integration: Requirements, state of the art and open challenges. *SIGMOD Rec.*, 41(1):11–19, April 2012. ISSN 0163-5808. doi: 10.1145/2206869.2206873. URL http://doi.acm.org/10.1145/2206869.2206873.

[23] H. Eldardiry, E. Bart, J. Liu, J. Hanley, B. Price, and O. Brdiczka. Multi-domain information fusion for insider threat detection. In *SPW*, pages 45–51, May 2013. doi: 10.1109/SPW.2013.14.

[24] Michael D Ernst. Static and dynamic analysis: Synergy and duality. In *WODA*, pages 24–27. Citeseer, 2003.

[25] C. Falge, B. Otto, and H. ÃŰsterle. Data quality requirements of collaborative business processes. In *HICSS*, pages 4316–4325, Jan 2012. doi: 10.1109/HICSS.2012.8.

[26] S. Fickas, G. Kortuem, and Z. Segall. Software organization for dynamic and adaptable wearable systems. In *Digest of Papers. First International Symposium on Wearable Computers*, pages 56–63, Oct 1997. doi: 10.1109/ISWC.1997.629920.

[27] Lori Flynn, Greg Porter, and Chas DiFatta. Cloud service provider methods for managing insider threats: Analysis phase ii, expanded analysis and recommendations. 2014.

[28] Michael Gertz and Sushil Jajodia. *Handbook of database security: applications and trends*. Springer, 2007.

[29] Saverio Giallorenzo. *Real-World Choreographies*. PhD thesis, Università degli studi di Bologna, 2016.

[30] Saverio Giallorenzo, Andrea Melis, and Marco Prandini. Smart Mobility for All. Technical report, University of Bologna, 2016.

[31] Henry G Goldberg, William T Young, Alex Memory, and Ted E Senator. Explaining and aggregating anomalies to detect insider threats. In *HICSS*, pages 2739–2748. IEEE, 2016.

[32] Paul Groth, Michael Luck, and Luc Moreau. A protocol for recording provenance in service-oriented grids. In *PDS*, pages 124–139. Springer, 2004.

[33] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future generation computer systems*, 29(7):1645–1660, 2013.

[34] Zygmunt J. Haas, Joseph Y. Halpern, and Li Li. Gossip-based ad hoc routing. *IEEE/ACM Trans. Netw.*, 14(3):479–491, June 2006. ISSN 1063-6692.

[35] Lianlian He, Peng Yue, Liping Di, Mingda Zhang, and Lei Hu. Adding geospatial data provenance into SDI âĂŤ a service-oriented approach. *AEORS*, 8(2):926–936, 2015.

[36] Thomas S Heydt-Benjamin, Hee-Jin Chae, Benessa Defend, and Kevin Fu. Privacy for public transportation. In *IWPET*, pages 1–19. Springer, 2006.

[37] Shuyuan Mary Ho, Jeffrey T Hancock, Cheryl Booth, Mike Burmester, Xiuwen Liu, and Shashank S Timmarajus. Demystifying insider threat: Language-action cues in group dynamics. In *HICSS*, pages 2729–2738. IEEE, 2016.

[38] Jan Holler, Vlasios Tsiatsis, Catherine Mulligan, Stefan Avesand, Stamatis Karnouskos, and David Boyle. *From Machine-to-machine to the Internet of Things: Introduction to a New Age of Intelligence*. Academic Press, 2014.

[39] Vincent C Hu, David Ferraiolo, Rick Kuhn, Arthur R Friedman, Alan J Lang, Margaret M Cogdell, Adam Schnitzer, Kenneth Sandlin, Robert Miller, Karen Scarfone, et al. Guide to attribute based access control (abac) definition and considerations (draft). *NIST Special Publication*, 800(162), 2013.

[40] Jeffrey Hunker and Christian W Probst. Insiders and insider threats-an overview of definitions and mitigation techniques. *JoWUA*, 2(1):4–27, 2011.

[41] Ashish Kamra, Evimaria Terzi, and Elisa Bertino. Detecting anomalous access patterns in relational databases. *The VLDB Journal*, 17(5):1063–1077, 2008.

[42] Miltiadis Kandias, Nikos Virvilis, and Dimitris Gritzalis. *The Insider Threat in Cloud Computing*, pages 93–103. Springer, 2013. ISBN 978-3-642-41476-3. doi: 10.1007/978-3-642-41476-3_8. URL http://dx.doi.org/10.1007/978-3-642-41476-3_8.

[43] Stuart E. Madnick, Richard Y. Wang, Yang W. Lee, and Hongwei Zhu. Overview and framework for data and information quality research. *J. Data and Information Quality*, 1(1):2:1–2:22, June 2009. ISSN 1936-1955. doi: 10.1145/1515693.1516680. URL http://doi.acm.org/10.1145/1515693.1516680.

[44] Sunu Mathew, Michalis Petropoulos, Hung Q Ngo, and Shambhu Upadhyaya. A data-centric approach to insider attack detection in database systems. In *RAID*, pages 382–401. Springer, 2010.

[45] Silvia Mirri, Andrea Melis, Catia Prandi, and Marco Prandini. Crowdsensing for smart mobility through a service-oriented architecture. In *ISCC*, page 5. IEEE, 2016.

[46] Silvia Mirri, Andrea Melis, Catia Prandi, and Marco Prandini. A service-oriented approach to crowdsensing for accessible smart mobility scenarios. In *Proceedings ICCTS*, page 5. IEEE, 2016.

[47] Silvia Mirri, Andrea Melis, Catia Prandi, and Marco Prandini. A microservice architecture use case for persons with disabilities. In *CVSJ*, page 5. Hindawi, 2016.

[48] Bhoomika R Mistry and Amish Desai. Privacy preserving heuristic approach for association rule mining in distributed database. In *ICIIECS*, pages 1–7. IEEE, 2015.

[49] David Molnar, Benjamin Livshits, Patrice Godefroid, and Prateek Saxena. Automatic context-sensitive sanitization, November 25 2014. US Patent 8,898,776.

[50] Fabrizio Montesi. *Choreographic Programming*. PhD thesis, IT University of Copenhagen, 2013.

[51] A. Moser, C. Kruegel, and E. Kirda. Limits of static analysis for malware detection. In *ACSAC*, pages 421–430, Dec 2007. doi: 10.1109/ACSAC.2007.21.

[52] Michel Mouly, Marie-Bernadette Pautet, and Thomas Foreword By-Haug. *The GSM system for mobile communications*. Telecom publishing, 1992.

[53] Hyeran Mun, Kyusuk Han, Chanyeob Yeun, and Kwangjo Kim. Yet another intrusion detection system against insider attacks. *Proc. of SCIS*, 2008.

[54] Mehmet Ercan Nergiz, Maurizio Atzori, and Chris Clifton. Hiding the presence of individuals from shared databases. In *SIGMOD*, pages 665–676. ACM, 2007.

[55] Nicola Nostro, Andrea Ceccarelli, Andrea Bondavalli, and Francesco Brancati. Insider threat assessment: A model-based methodology. *ACM SIGOPS*, 48(2):3–12, 2014.

[56] Eugen Petac, Andreea-Oana Petac, et al. About security solutions in fog computing. *Ovidius University Annals, Economic Sciences Series*, 16 (1):380–385, 2016.

[57] S. Pippuri, S. Hietanen, and K. PyyhtiÃď. Maas finland. http://maas.fi/.

[58] Benny Rochwerger, David Breitgand, Eliezer Levy, Alex Galis, Kenneth Nagin, Ignacio M Llorente, Ruben Montero, Yaron Wolfsthal, Erik Elmroth, Juan Caceres, et al. The reservoir model and architecture for open federated cloud computing. *IBM JRD*, 53(4):4–1, 2009.

[59] Rodrigo Roman, Jianying Zhou, and Javier Lopez. On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, 57(10):2266–2279, 2013.
[60] Rosario Salpietro, Luca Bedogni, Marco Di Felice, and Luciano Bononi. Park here! a smart parking system based on smartphones' embedded sensors and short range communication technologies. In *Internet of Things (WF-IoT), 2015 IEEE 2nd World Forum on*, pages 18–23. IEEE, 2015.
[61] Ravi Sandhu. Attribute-based access control models and beyond. In *ASIACCS*, page 677, 2015.
[62] Ravi S Sandhu, Edward J Coynek, Hal L Feinsteink, and Charles E Youmank. Role-based access control models yz. *IEEE computer*, 29(2): 38–47, 1996.
[63] Bob G Schlicher, Lawrence P MacIntyre, and Robert K Abercrombie. Towards reducing the data exfiltration surface for the insider threat. In *HICSS*, pages 2749–2758. IEEE, 2016.
[64] Bruce Schneier. *Secrets and lies: digital security in a networked world*. John Wiley & Sons, 2011.
[65] Asaf Shabtai, Yuval Elovici, and Lior Rokach. *A survey of data leakage detection and prevention solutions*. Springer, 2012.
[66] Nahla Shatnawi, Qutaibah Althebyan, and Wail Mardini. Detection of insiders misuse in database systems. In *ICECS*, volume 1, 2011.
[67] George Silowash, Dawn Cappelli, Andrew Moore, Randall Trzeciak, Timothy J Shimeall, and Lori Flynn. Common sense guide to mitigating insider threats 4th edition. Technical report, DTIC Document, 2012.
[68] Yogesh L Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance techniques. *CSD, IU, Indiana*, 47405, 2005.
[69] Yogesh L. Simmhan, Beth Plale, and Dennis Gannon. A survey of data provenance in e-science. *SIGMOD Rec.*, 34(3):31–36, September 2005. ISSN 0163-5808. doi: 10.1145/1084805.1084812. URL http://doi.acm.org/10.1145/1084805.1084812.
[70] Lance Spitzner. Honeypots: Catching the insider threat. In *CSAC*, pages 170–179. IEEE, 2003.
[71] Vasilis Stavrou, Miltiadis Kandias, Georgios Karoulas, and Dimitris Gritzalis. *Business Process Modeling for Insider Threat Monitoring and Handling*, pages 119–131. Springer International Publishing, Cham, 2014. ISBN 978-3-319-09770-1. doi: 10.1007/978-3-319-09770-1_11. URL http://dx.doi.org/10.1007/978-3-319-09770-1_11.
[72] Wei-Tek Tsai, Xiao Wei, Yinong Chen, Ray Paul, Jen-Yao Chung, and Dawei Zhang. Data provenance in SOA: security, reliability, and integrity. *SOCA*, 1(4):223–247, 2007.
[73] Vijaya Bhaskar Velpula and Dayanandam Gudipudi. Behavior-anomaly-based system for detecting insider attacks and data mining. *IJRTE*, 1(2):261–266, 2009.
[74] K. Xu, H. Xiong, C. Wu, D. Stefan, and D. Yao. Data-provenance verification for secure hosts. *DSC*, 9(2):173–183, March 2012. ISSN 1545-5971. doi: 10.1109/TDSC.2011.50.
[75] Pengfei Zhou, Yuanqing Zheng, and Mo Li. How long to wait?: predicting bus arrival time with mobile phone based participatory sensing. In *Proceedings of MobiSys*, pages 379–392. ACM, 2012.