DOCTORAL SCHOOL
MEDITERRANEA UNIVERSITY OF REGGIO CALABRIA

DEPARTMENT OF INFORMATION ENGINEERING, INFRASTRUCTURES
AND SUSTAINABLE ENERGY
(DIIES)

PHD IN
INFORMATION ENGINEERING

S.S.D. ING-ING/05
XXIX CYCLE

ORGANIZATIONS AND COMMUNITIES: TRUST, SECURITY, AND PRIVACY ISSUES

CANDIDATE
SERENA NICOLAZZO

ADVISOR
Prof. Francesco BUCCAFURRI

COORDINATOR
Prof. Claudio DE CAPUA

REGGIO CALABRIA, FEBRUARY 2017

Finito di stampare nel mese di Aprile 2017

Edizione Centro Stampa d'Ateneo

Collana *Quaderni del Dottorato di Ricerca in Ingegneria dell'Informazione*
Curatore *Prof. Claudio De Capua*

ISBN 978-88-99352-22-6

Università degli Studi *Mediterranea* di Reggio Calabria
Salita Melissari Feo di Vito. Reggio Calabria

SERENA NICOLAZZO

ORGANIZATIONS AND COMMUNITIES: TRUST, SECURITY, AND PRIVACY ISSUES

Discere ne cesses; cura sapientia crescit;

Rara datur longo prudentia temporis usu.

(Disticha catonis 4,27)

Preface

This book is my PhD thesis and presents the research work I did at the DIIES department of the University *Mediterranea* of Reggio Calabria during the XXIX PhD course cycle from 2013 to 2016, under the supervision of Prof. Francesco Buccafurri.

My research activities were carried out, and are currently carried out, in cooperation with high experienced and professional researchers such as Francesco Buccafurri himself, Gianluca Lax and Antonino Nocera from whom I had the opportunity to learn everything I needed to complete the research work described in this thesis.

My research is based on the observation that, in this era of great technological evolution, the notion of physical space is growing complex by including also the virtual dimension. As a consequence, human interactions with other humans and the surrounding environment are changing strengthening the impact of virtual communities over those that meet physically.

This scenario opens new opportunities for both research and business. Indeed, the knowledge coming from the study of virtual communities along with information about community members themselves, can feed advanced analysis, whose results are extremely valuable for a number of applications for physical communities. However, as the use of this knowledge becomes pervasive, an always increasing number of security and privacy aspects must be taken into account. Access policies, data storage, and security of data transfer, are prominent examples in this setting. Therefore, in such a complex scenario, an integrated vision and cross dimension analyses become mandatory.

This thesis follows the above reasoning, by addressing some important trust, security, and privacy issues in both dimensions: online communities and physical organizations. According to this choice, it can be organized into two main macro-areas. The former covers our attempts to investigate the aspects related to privacy, security and trust in online communities. The latter macro-area presents models and approaches to handle privacy and security on both public and private physical organizations.

It is a pleasure to thank the people who have helped me during this work. Firstly, I would like to express my gratitude to my advisor Prof. Francesco Buccafurri for the support of my PhD studies and related researches. His guidance helped me in my research activity during my PhD course and in writing this thesis.

Besides my advisor, I would like to thank the rest of the computer engineering group of the DIIES department who supported me during these years: Prof.Gianluca Lax, Dr. Antonino Nocera, especially for their insightful comments, encouragements and the time spent together.

February 2017 Serena Nicolazzo

To my family, especially to my aunt Giusy
Alla mia famiglia, a mia zia Giusy

Contents

List of Figures

List of Tables

1

Introduction

This chapter is devoted to describe the framework in which this thesis lies and to introduce the issues here faced. The chapter is divided into four main parts. The first gives a brief premise, the second part introduces the aspects related to privacy, security and trust in online communities. The third part presents the results achieved in the fields of privacy and security on both public and private physical organizations. Finally, in the last section, we provide an overview of the thesis organization.

1.1 Premise

In this era of disruptive technological evolution, the environment around us is becoming something that spans multiple dimensions, expanding beyond the notion of physical space. Accordingly, also human interactions (both among humans and with the environment) are changing. Virtual communities are as real as communities that meet physically and, in most cases, the edge of the virtual and the real is blurred.

This scenario opens new opportunities for both research and business. As a matter of fact, virtual communities are places where people with common interests share knowledge. This knowledge, along with information about community members themselves, can be extremely valuable to perform analysis, whose results can provide many benefits for a number of applications built on top of physical communities. Some interesting and up-to-date examples can be the physical access control to critical environments, and the formalization of tools to provide secure and traceable transactions among people. However, there are a number of security and privacy implications to take into account, including data access policies, data storage, security of data transfer, and so on. As a consequence, an integrated vision is needed, and cross dimension analyses are welcome.

This thesis follows the above trace, by addressing some basic yet important trust, security, and privacy issues in both dimensions: online communities and physical organizations. This leads to divide the thesis into two main macro-areas, which are introduced in the following two sections.

1.2 Online Communities

Online Social Networks (OSNs, for short), such as Twitter or Facebook, have experienced exponential growth in the last decade. Although these social systems offer powerful means for interaction and communication and attracted the interest of many researchers from disparate fields, they also give raise to privacy and security concerns. The objective of the first part of this thesis is to deeply study some aspects related to privacy and security of online communities by using OSNs as reference models for this investigation.

However, because nowadays users spread their contents throughout several social systems, the power of OSNs can be fully exploited only if we move from a single-social-network to a multi-social network perspective, still keeping a user-centered vision. Indeed, despite the conceptual uniformity of the social-network universe in terms of structure, basic mechanisms, main features, etc., each social network has, in practice, its own terms, resources, and actions. This is a strong handicap for the design and implementation of applications aiming at analysing different aspects of online communities. As a matter of fact, little exists in terms of models and languages to support social-network-based programming in large, according to software engineering principles of genericity and polymorphism.

For this reason, a first goal of this thesis is the implementation of a model to generalize and match concepts, actions and relationships of existing social networks [40, 52]. Clearly, our aim is not just the development of some APIs working over all social networks, but of an approach allowing us to keep the typical semantics structure of a social network in this new multi-social network perspective. From this point of view, a user-centered vision assumes a crucial role because, besides maintaining all entities and relationships of single social networks, allows us to transparently associate with a user all the information coming from the set of social networks he belongs to.

This model is used throughout this thesis as an important tool to uniformly handle social network data and as a mean to accomplish the crucial task of extracting data from a multi-social network scenario to retrieve all the information needed for our study on privacy and security on OSNs [46].

Specifically, concerning our investigations on privacy, we focus on two important aspects: *(i)* the analysis of user behavior when it comes of privacy and disclosure of personal information w.r.t. friendship and user activity; *(ii)* the robustness of social network privacy settings. This investigation can bring significant results useful for different entities: for Internet and OSN providers to guide infrastructural and application-level actions, for users themselves to enhance awareness in this potentially insecure world, for companies and government institutions to make better use of this huge network of people for their finalities, for scientists to better understand individuals and communities.

Clearly, in this case, the trivial comparison of the behavior of (different) users on different social networks does not give correct information, so we cannot just elaborate results obtained in the literature in the different social networks. To give a trivial example, if we want to study how the behavior of drivers change in cars A and B, we should study a sample of people driving both the cars, and observe the differences in the two experiences. We cannot simply study the expected behavior of drivers of car A and the expected behavior of drivers of car B and compare them, because the result would be affected by those traits that, for example, pushed people to use car A instead of car B. The same happens for comparative studies on behavioral aspects of online social networks, leading to the necessity of considering membership overlap (i.e., users belonging to all the studied OSNs) as the right perspective from which drawing meaningful and well-founded results. Therefore, we focus on the activities of the same user in different social networks by following the above multi-social network perspective. Our aim is to study a number of behavioral aspects of people in social networks and to use this knowledge to further investigate privacy concerns and user habits in term of disclosure of personal information. We perform this study on the two most popular social networks in the current Web scenario, which are Facebook and Twitter.

We start this analysis by studying friendship relationships. Indeed, OSNs are important for maintaining social relations and previous studies have found that friendship in such systems is positively correlated with many other aspect of both virtual and real life. As for this aspect, we study what is the attitude of users to have friendship relations overlapping between Facebook and Twitter and if a correlation between the number of friends in Twitter and Facebook exists.

Then, we move our focus to the analysis of the activity level of users belonging to both Twitter and Facebook. [193] found that the prime goal of user activity on Facebook is to self-promote or to maintain relationships, whereas other studies showed that some types of activity are a sign of narcissism [203]. Our study aims to answer the question "What about user activity and how the prevalence of activity on Facebook or Twitter is correlated to membership overlap?".

The knowledge derived by the former study is then used to investigate privacy concerns and the disclosure of personal information in relation to the user activity level and the membership to more social systems. Indeed, recent research results on Facebook have shown that both a strong association between low engagement and privacy concern [220] and a significant relationship between privacy awareness and privacy concerns/self-disclosure [255] exist. Our study aims at answering the question "Is there a connection between user awareness about privacy threats, user activity level and the membership overlap between Twitter and Facebook?".

The answers to all these questions are interesting and sometimes surprising, however a basic conclusion of this study is that there is an always increasing number of active users having more online privacy literacy and that the use of privacy tools provided by social networks will be more and more widespread and critical in the next future. This consideration is the motivation of a subsequent investigation focusing on security in OSNs and, specifically, on the robustness of the privacy protection feature of Facebook.

Facebook's privacy setting gives users the possibility to choose who is allowed to see their profile information. Hence, a user who does not want to reveal his friend list information to everyone can specify to hide such an information in the privacy setting page. By default, everyone can see the friends of a user. However, there are many Facebook users having a private list of friends, meaning that this information is often (reasonably) perceived as sensible. We show that this privacy protection feature can be broken even in the least advantageous conditions for the adversary [49].

Obviously, if the adversary knows the victim in the real life or has information about the contexts in which the victim lives, he can easily guess some Facebook profiles owned by real-life friends of the victim whose friend list is public. It is rather intuitive that only few seeds are enough to discover incrementally large portions of private friends, as usually friends form highly connected clusters. Therefore, this case is trivial. But we want to consider the most difficult case. The adversary has only the name of the victim and the link to his Facebook profile, he can guess only some general information about him (nationality, for example), but has no information about his real life, his job, his interests, etc. This case, for example, may occur in preliminary cyber investigations, if the Web intelligence does not have the complete access to Facebook database. Again, the privacy of the list of friends can be broken once only a few friends (even one) are found with public profiles. But, how to find them? Since guessable general information selects a very large portion of Facebook users, it would seem that the only way for the adversary is to try an infeasible guess-and-check attack. However, once again, we adopt a multi-social network view and we leverage a new social property, called "Interest Assortativity", to build a much more efficient attack allowing the adversary to break the privacy of the victim in most of the cases. The "Interest Assortativity" property is also described in this thesis and is another interesting output of this work that can find application in a lot of desperate scenarios [51].

From what has been stated before, since their development, OSNs have played a continuous role in how people interact with each other. Moreover, they contribute to how companies interact with potential customers as well. This is even more true when it comes of mobile applications based on social network services. Indeed, the diffusion of smartphones has rapidly modified people habits, by allowing people to be always connected via social networks, but also by enabling ubiquitous and pervasive applications. This has dramatically enlarged the attack surface making each user of mobile applications a potential victim of cyber-attacks. Users can be affected in several ways, including data theft or corruption, annoyance, device damage or location tracking.

Among mobile applications, social network applications have a great potential for development, because they use social networks as a platform for information sharing, user-centered content production, and interoperability. Activities and interests can be shared among people, making applications aware of the social factor, which is often determinant for the success and the effectiveness of an application.

However, one of the problems that can limit the full diffusion of social network applications is that, in the current real-life scenario, most of social network providers do not support a fine-grained control when an application uses APIs of social networks to access a profile. This is the case of Twitter, in which the supported access control policy is roughly on/off. For instance, if an application needs the right to tweet within a given user profile, the user is enforced to grant to the application the right to modify (even to delete) any information included in his profile with no restriction. This enables a large set of security threats because users cannot fully control what their applications do in their Twitter profile, so they cannot block their potential malicious behavior.

We can argue that the effect of the above policy may strongly limit the growth and the diffusion of social network applications, despite their strategic utility. For instance, even for an inexpert user, it can appear inopportune to run an application asking for a complete control of his Twitter profile.

To overcome this problem, we propose an effective solution working for Android Twitter applications based on a middleware approach [47]. This middleware is spread-out between the smartphone operating system and a server-side platform, and allows the definition of a fine-grained access control model to protect end-users. The client-side middleware is thought as an application that *hijacks* Twitter API calls to the server-side middleware, where calls are implemented according to the access control rules. However, a large-scale adoption of our solution could be implemented as an extension of Android itself. Interestingly, our solution enables other possible benefits, as anomaly-based malware detection leveraging API-call patterns. Moreover, by using the aforementioned model thoroughly described in this thesis, it can be extended to a multi-social network scenario.

The last topic we deal with in the context of online communities is related to trust and reputation. The use of trust-based approaches in real-life systems has been widely recognized as a promising solution to improve the dependability of these systems. The term *reputation* is used to measure the indirect trust perceived by a community with respect to a given service provider. Reputation has thus a crucial role when a user u does not have sufficient knowledge about a service provider s because of the lack of past experience. Then, u can use the reputation of s to decide whether to use services provided by s.

There are various models to represent reputation (often combined with *reliability*, which is a measure based on direct past experiences). However, there are many open problems mainly related to the issue of feedback trustworthiness. As a matter of fact, the reputation measure suffers from an intrinsic weakness related to the competitive nature of systems where it is used, so that a user cannot always assume that other users will behave honestly. Some actions could be done with the goal of deceiving the community to obtain some advantage. Trust-based approaches tend to keep the system dependable also in this case.

In the literature, the notion of *certified reputation* has been introduced in the context of open Multi-Agent Systems (MAS) [123]. However, the translation of the general concept into concrete models is a challenging and still open issue.

This part of this thesis gives a contribution in this research field, by proposing a reputation model that abstractly considers service providers, users and feedbacks, and implements the theoretical notion of certified reputation to concretely define a strategy to *normalize* feedback scores towards reliable values [42, 44].

We apply this model to the case of TripAdvisor, which is a very famous travel website collecting reviews of travel-related contents. On the basis of these reviews, an aggregative score of each content is shown. Our approach can be profitability applied to TripAdvisor, as we cannot assume that all its actors behave honestly and thus the difference between user evaluations derives not only from physiological subjectivity. Instead, there are many users who try to jeopardize the reputation system by means of malicious behavior for vandalism or, more likely, to obtain advantages. On the other hand, TripAvisor (as other similar services) is becoming more and more a system with critical impact, as from it may depend the success or the end of business as hotels and restaurants. Therefore, improving its dependability is certainly a relevant challenge.

We propose a solution to the above problem not increasing invasiveness nor reducing usability of the system. Moreover, our approach fully guarantees backward compatibility, which is an essential requirement in solutions aimed at a real utilization.

1.3 Physical Organizations

The second part of this thesis focuses on strategies assessing privacy and security on both public and private physical organizations.

Nowadays, the digitalization of the public sector and its services is increasing. This trend, known as e-government, leads to changes in the way communications between people and the public sector happen, and it has impact also on the way the public sector is governed and on how public services are delivered.

Clearly, these changes give raise to a number of challenges in term of security and privacy. Indeed, a basic level of confidence and security must be established in order for people to trust and use digital public services.

One of the main problems in this context is the authentication of transactions. The typical signature formats adopted for qualified electronic signatures are CADES and XADES signatures [1]. Qualified electronic signature has become the basic tool of any digitalization process, where exchanging documents with full legal validity has a significant role. In general, we expect that both in e-government applications and in transactions between citizens and companies, the use of qualified electronic signature will always be increasing in the next future.

However, in this process, there are some aspects to take into account. These aspects are of two types: related to the cost and related to the usability of qualified electronic signature. Indeed, the cost of qualified signature creation devices (smart cards or HSM-services) is certainly not negligible. Moreover, the invasiveness of the operations related to signature creation, signature verification, user registration, and certificate management is relevant. These aspects are under the attention of legislators. For instance, the European Union has recently adopted an act [19] which promotes the use of the advanced electronic signature, for which no qualification of certificates and devices is required. The new regulatory framework allows the legal enforceability of advanced electronic signature, for which advanced electronic signature has automatic legal value if applied to closed domains, such as document exchange between municipal public offices and registered citizens, university and its students, or private company and employees.

This framework gives a strong motivation to design new protocols that, besides the specific features of qualified electronic signature, relax also the use of public-key encryption. As a matter of fact, advanced electronic signature does not require a specific technological solution too, even though no concrete solutions not using public-key encryption are currently adopted.

In this context, we propose a new advanced electronic signature protocol not using neither public-key encryption nor qualified signature creation devices nor qualified certificates [38, 59, 39]. The basic idea underlying our solution is to leverage the following properties: (1) the power of online social network to share information among people, (2) the plausibility of a number of assumptions concerning the dependability of services provided by online social networks, (3) the trustworthiness of social network providers, and (4) the level of assurance of the accountability associated with the interactions between users and their social network profile. The signature functions are spread out over social-network profiles, by giving to posting and searching operations a central role in both signature generation and signature verification processes in a very usable and scalable fashion. This makes our signature strongly oriented to cooperative work and information-sharing-based applications. To be realistic and effective, we implement our solution on the popular social network `Twitter` without assuming any change of its features and through a deep security analysis, we prove its resistance to known attacks in the context of qualified electronic signature.

Always in the context of e-government, another critical aspect is related to the authentication of users in scenarios where their privacy is threatened by *honest-but-curious* providers. This aspect is extremely timely in the context of cloud computing, in which the knowledge available to providers about services accessed by their users may be an important mean of privacy leakage.

Cloud computing is recently receiving a lot of attention from both research and industrial worlds. The cloud paradigm allows a user to transparently move his storage and computation to servers distributed all over the Internet (i.e., cloud) that implement services on-demand. Cloud provides its customers with reliable, efficient, and low cost computing services such as e-mail, instant messaging, storage systems, etc. However, such an outsourcing paradigm introduces new security and privacy threats, mainly related to the fact that cloud providers become owner of (even sensitive) information regarding their customers. In application contexts linked to e-government, privacy requirements become crucial, indeed sensitive information can be drawn by just observing which services a user is accessing to, even though we assume that contents are fully obscured. Indeed, it is widely accepted that the adoption of an *honest-but-curious* adversary assumption can be realistically done in the context of cloud providers. In fact, information regarding customers may give these parties strong strategic advantages. Thus, even if we could assume that cloud providers execute services correctly, they might look at the information exchange between entities.

While the aspect of data confidentiality and the related issue of key management have received a lot of attention in the recent scientific literature, the problem of information leakage arising from the observation of user requests (i.e., accesses to cloud services) has been much less investigated. Although a number of proposal leveraging anonymous authentication schemes, group signatures, zero knowledge protocols exist [72, 169], a number of challenging problems should be completely addressed to make these solutions really applicable.

Consider, for example, the case in which a government party has the role of end-service provider offered to citizens through non-government cloud providers. This is an emergent scenario, due to the general difficulty of governments to adopt national clouds. In this case, customers of a cloud can operate promiscuously, both for e-government and private services. Here, an opportunity arises. A trusted third party exists for free (e.g., some e-government entity), which can play a role in the authentication process (consider that we are assisting to a rapid evolution of EU Countries towards digital identity systems [13]), and we may assume that no collusion with cloud providers exists. At the same time, besides the strong requirement of privacy against cloud providers, a specific issue appears. A full accountability of all user activity is necessary if we consider both the responsibilities coming from law requirements and the general need of security.

In this reference scenario, we propose a solution providing anonymous access to cloud services yet preserving full user accountability and showing nice characteristics of computational cheapness [45]. With no collusion of the involved parties, no information about the identity of the user accessing the cloud is obtainable. Accountability is guaranteed, in case of need, by merging information coming from multiple parties.

Importantly, anonymity of user activity is reached by guaranteeing both the anonymous authentication and the unlinkability of user requests. To do this, we combine a multi-party cryptographic protocol with a cooperative P2P-based approach. We believe that this new way to integrate P2P and cloud computing, in which the customers of the cloud cooperate with each other to obtain privacy features and increased efficiency, is sustainable also from a business point of view, due to the reciprocal advantage obtained by users. Conversely, a solution based on Tor [139], appears not realistic due to legal problems which the subscription to such anonymization system may result in.

Another interesting and new aspect we focus on is Blockchain technology, that allows mutually distrustful parties to transact safely without trusted third parties and avoiding high legal and transactional cost. Coordination, record keeping, and irrevocability of transactions are features that make Blockchain technology exploitable not just for crypto-currencies. Indeed, a variety of applications can be built on top of this technology, making Blockchain a registry and inventory system for the recording, tracking, monitoring, and transacting of all assets.

Despite the power of Blockchain whose charm is recently attracting the interest of many companies and researches throughout the world, it is well-known that the concrete implementation of the protocol is not free from weaknesses. The first is that the real Blockchain nodes are only full nodes, the other nodes must trust (some) full nodes. Thus, the full P2P-like decentralization is only theoretical. Coherently, sybil attacks on non-full nodes could be performed to deceive them in the verification step. Another serious problem of Blockchain is that its security is based on the *proof of work*, which is very expensive from a computational point of view and then also from a global energy consumption point of view. Consider that the proof of work results in a highly duplicated computational efforts, of which only a small fraction is useful for the protocol. In other words, the protocol determines a huge quantity of work, and thus energy lost. The last unpleasant feature of Blockchain is that any participant in the Blockchain system, even a user, must enter in a P2P network, and this is not well-accepted from many people, even for the perception of low security.

Starting from the above observations, we address the following question. Is there an alternative to Blockchain? We found an answer in the extraordinary power of OSNs in sharing information among people. Having in mind that this power should be exploited in a richer and more complex way than the state of the art, we propose a fully decentralized protocol that mimics Blockchain transactions by simple tweets, thus solving natively the problem of agreeing a common transaction sequence, from which the need of the proof of work arises in Blockchain. We call this protocol `Tweetchain` [56, 57]. The transaction tweets, together with a sufficient number of confirmation tweets given by the community, form a meshed chain making impossible for anyone, including Twitter itself, to alter, delete or forge valid transactions. All nodes participate in the same way, so that from a conceptual point of view, the protocol is truly decentralized. No P2P feature must be enabled on the client machine, but only an application working on the Twitter profile of the corresponding user. The role of Twitter is not that of a trusted third party. Not even a provider of some part of the protocol. It just continues to publish tweets, as before. Importantly, Twitter cannot selectively omit the collaboration on some transaction, some user, some period, etc. It could only block the whole Tweetchain community. This is very little plausible and is similar to the event that Blockchain is stopped by the entities providing the Internet. Anyway, to prevent this possibility, we could extend our protocol to involve multi-social networks by using, once again, the model described above.

In the context of physical organization, besides investigating approaches related to e-government, another fundamental activity concerns the definition of strategies to improve the security of physical environments. Therefore, this part of the thesis deals also with issues related to the field of urban and homeland security.

Specifically, urban and homeland security are settings in which monitoring systems produce a huge amount of data, and where the integrated vision mentioned earlier is more and more important. An emblematic case is that of video surveillance, especially if a capillary solution is adopted. Since, as stated before, more and more often, both private and public organizations rely on cloud-based services supplied by third parties, it is plausible that the municipality finds convenient to move video surveillance data to a third-party cloud. In this case, the trustworthiness of the cloud provider becomes a critical point. Among other threats, the possibility that the cloud does not return intact responses to queries has to be considered. In order to solve this issue, we propose an approach to allow the verification of the completeness, freshness and correctness of cloud query results.

This is a very important feature as, nowadays, a user has often the (legal) necessity of proving that query results are not compromised.

This problem is well known in the literature, under the name of *query integrity*, and the increasing attention towards the cloud has also renewed the interest in this issue [209]). Solving this problem means to allow users to verify that query results are *complete* (i.e., no qualifying tuples are omitted), *fresh* (i.e., the newest version of the results are returned), and *correct* (i.e., the result values are not corrupted).

In our scenario of video surveillance data streams, append operations and range queries are dominant, and the efficiency is a critical factor. Our focus is mainly about the high frequency of the insert operation (which is, in this case, an append operation), and about the limited resources in terms of both computation and power of devices responsible of insertions in a distributed environment.

In a similar scenario, classical general-purpose deterministic techniques for query integrity appear little suitable, as the computation required to update the extra data structures could become a bottleneck. To overcome this drawback we propose a new deterministic technique, which is proved to be more efficient on insertions than state-of-the art techniques, and also efficiently supporting range queries, which are the relevant queries in this setting (a user typically looks for video temporal intervals) [50, 53, 54, 55]. Therefore, our ambition is certainly not to invalidate the state of the art on the general-purpose deterministic techniques for query integrity, but just to address this problem in an emergent application and very timely in the context of urban security (we are more and more moving towards distributed *sensing* low-power-device architectures) and to find a suitable solution to enable query integrity verification.

Clearly, monitoring systems based on video surveillance could be little suitable in contexts where people privacy could be violated by the continuous recording of their movements and, in most of these cases, such a solution could not be compliant with law requirements. As a consequence, finding the right trade-off between people privacy and security/safety requirements is an important challenge.

Therefore, another interesting aspect in the context of urban security we deal with in this thesis is the design of a complex system based on a probabilistic framework which supports a re-interpretation of the classical *k-anonymity* notion [48, 41, 43, 39]. To implement this model allowing us to guess patient location with probability k^{-1}, we leverage an infrastructure of RFID readers covering an areas of the monitored environment, in which people are equipped with suitable designed RFID tags. RFID tags periodically send a (PRNG-based) dynamic ID that does not disclose patient's identity because it is always guaranteed that at least other $k-1$ patients send the same ID at the same stage. This probabilistic framework allows us to tune the parameters of the proposed strategy to solve the trade-off between localization precision and privacy level. Obviously, whenever $k = 1$, our model reduces to existing people' localization systems from the side of precision, but maintains privacy guaranteed against both malicious access of an intruder to location data and eavesdropping attacks.

We study two application scenarios leveraging this framework: the first is related to critical infrastructures, whereas the second deals with Assistive Living Facilities (ALFs).

The first application deals with critical infrastructure. The problem of accountability of people's access to physical environments is receiving great attention in recent times, also because of the emergent risks related to terrorism. Specifically, indoor tracking to allow a-posteriori identification of the authors of an illegal action is an important issue. Consider, for example, a security/safety incident occurred in a museum, which hosts thousands of visitors per day. Besides video surveillance, an effective possibility for tracking people is based on the use of RFIDs, in such a way as to have logs reporting people localization at any time, allowing us to restore precise and decisive information in a faster way than video surveillance. This kind of solution requires that people entrance in the surveillance environment is registered. Anyway, this is an acceptable requirement (often already adopted) in case of critical environments (government buildings, museums, safety-critical environments, tribunals, etc.) and high-level attention.

Clearly, in this scenario the application of the probabilistic framework based on the *k-anonymity* scheme described above may be suitable. Indeed, the requirement of storing a continuous log on the accesses to all the physical entrance in a building gives rise to other security challenges.

To solve this issue, we propose an RFID/BeagleBone-based system leveraging our *k-anonymity*-based framework to generate logs in which events are stored in such a way that the association with people is possible with a suitable degree of uncertainty. More in details, generated logs fulfill the *k-anonymity* property for which we are able to guess who accessed a place (for example, a room of a museum), at a given time, with probability k^{-1}. Indeed, the server receives *k-anonymous* logs and no party knows detailed information. It is worth noting that this concept of *k-anonymity* meets the security requirement of identifying the person who was present in a given location at a given time, with uncertainty equal to k. This means for example that, in case of a security/safety incident (e.g., an explosion caused by an act of terrorism) we can restrict the pool of suspects from the whole (potentially huge) population to a few (i.e., k) people.

The above goal is obtained by running a distributed algorithm over the nodes of the network (i.e., the "smart" RFID readers) in such a way that they independently generate quasi-IDs guaranteeing the above privacy property. We remark that the distributed fashion of our methodology is the basis of its effectiveness from the point of view of privacy. Indeed, any centralized solution would not be able to protect data from attacks in which we assume that the adversary accesses the central server. In contrast, in our solution, only a simultaneous (unrealistic) access of the adversary to all the nodes of the network would allow him to break privacy.

As an evolution of the approach described above the second solution deals with Assisted Living Facilities. In health assistive environments the possibility of knowing patient's position can be very important, since may represent a valid support to medical activities and assistance strategies. Obviously, location data are strongly sensitive, especially in the context we are considering, so that we have to adopt suitable strategies in order to avoid that such an information can be maliciously used by unauthorized third parties. In some cases, an assistive environment includes regions so sensitive that it is questionable that even a trusted party has full control on patients' location data. Besides standard hospitals, this is the case, for example, of caring assistive living units, drug rehabilitation or alcoholism treatment centers, hospices and elderly care institutions, institutions for mental disorders, etc. As a matter of fact, a large effort is required by standards and norms to enforce the prevention of health privacy violations also by abuse of authorized parties. But, where is the exact border between utility of having precise information about patients' location and the right of keeping private the exact movements of patients is unclear. Thus, the solution of the above trade-off merits great attention from the scientific community. Indeed, commercial and research solutions, even based on past patents such as [3], exist aimed at positioning patients in health assistive environments. However, none of them stops misuse and curtails privacy violations also from the provider of the localization service (even due to data loss or theft). We argue that more advanced solutions of the above trade-off are desiderable, to make localization systems resistant to threats to patients' privacy which can prejudice human dignity and fundamental rights. Clearly, the solution to this problem is intrinsically very different with respect to that adopted for the access monitoring to critical areas. Indeed, here the system should be able to provide patients' location in every moment inside a given environment. It is worth noting that our contribution has some merits also from a theoretic point of view, as it introduce a novel perspective in the field of *k-anonymity* in location-aware applications. Indeed, while the classical *k-anonymity* localization aims to satisfy privacy requirements by changing or extending the exact user positions in such a way that *k* users are confused each other, our approach returns for each location a number of *k* possible users, with no detectable correlation. This new concept of *k-anonymity*, not only preserves privacy like in the classical case, fully meets the requirement of finding a user with a small number of attempts. This is an indefeasible feature in an assistive environment, in which a number of events requiring a rapid localization of a patient may occur. For example, knowing anytime just a small number of possible positions of a given patient living in a nursing home for elder adults or in a mental hospital, does not give us the possibility to infer his real position, but allows us to reach the patient through some additional information (like that obtained from the medical staff belonging to the candidate places, or by means of a physical search). Thus, the application of *k-anonymous* localization to the context of assistive environments does not regard the possibility of monitoring patients' moving behavior, but just the possibility of leaving the (critical) patient free of moving in a possibly huge environment without the risk of not to be able to quickly find him when needed (security reasons, therapy actions, assistive requests, behavior checks, emergency, etc.). A typical case when the above conditions fully occur is for patients with Alzheimer's. Observe that a malicious usage of localization queries aimed to track patients' moving behavior is contrasted in our approach simply by inhibiting continuous querying in such a way that two successive location queries cannot be related through the distance of the corresponding replies.

1.4 Plan of the Thesis

This thesis studies a number of problems related to trust, security and privacy within both virtual and physical communities. The structure of this thesis is composed of three parts. The first deals with the aspects related to privacy, security and trust in online communities and provides all details on tools and models built to carry out such investigation. The second part focuses on strategies assessing privacy and security on both public and private physical organizations. Finally, the third part presents the conclusions and future research directions.

More in detail, Chapter 2 deals with the crucial aspect of extracting data from a multi-social network scenario. To solve this task a model to match concepts in social networks and a framework providing support meta-APIs are presented. This model will be exploited throughout this thesis as a fundamental tool to manage data and information coming from this multi-social network environment.

Chapter 3 describes a comparative study on the behavior of users in two very popular social networks. The aim of this investigation is to understand how users deals with privacy and disclosure of personal information in relation to some measures based on user activity and friendship relationships. Moreover, this chapter focuses on the robustness of Facebook privacy settings and describes a potential threat to such a system.

Chapter 4 focuses on the security of social-network-mobile apps and, here, a middleware approach is proposed to allow the implementation of fine-grained access control rules to Application Programming Interfaces provided by social networks.

The last chapter of the first part, Chapter 5, discusses the topic of trust in online communities. It describes a reputation model abstractly implementing the theoretical notion of certified reputation by leveraging a large set of information (service providers, users and feedbacks) available in social networks. This model is, then, applied to a real case-study, i.e. to the TripAdvisor tourism platform.

The second part of this thesis, instead, deals with some strategies for privacy and security on both public and private physical organizations. The first three approaches described in this part lie in the field of e-governement. In particular, Chapter 6 presents a new electronic signature protocol, which does not use public-key encryption, qualified signature creation devices or qualified certificates. This approach leverages the power of online social networks and spreads out signature functions over social-network profiles.

In Chapter 7 we propose an authentication scheme supporting full anonymity of users and unlinkability of service requests in a cloud computing environment. Whereas, in Chapter 8 we present an alternative to Blockchain that leverages the popular social network Twitter, and ensures the security of transactions by building a meshed chain of tweets. The last three chapters of the second part of this thesis describe issues related to the field of urban security. In particular, Chapter 9 proposes an approach to allow the verification of the integrity of range queries in an environment with high constraints in terms of insertion computational costs. Chapter 10 and Chapter 11 present the application of a framework that allows the monitoring of people movements using a privacy-preserving identification technique, to two scenarios: Critical infrastructures (Chapter 10) and Assistive Living Facilities (Chapter 11).

Finally, in the third part of this thesis, Chapter 12 sketches some future developments of our research, whereas in Chapter 13 we draw our conclusions. The bibliography closes this part of the thesis.

Part I

Online communities

The role of Online Social Networks (OSNs, for short) is becoming increasingly important in everyday life. This is also true for research scenarios, where OSNs are considered as reference models to analyse communities.

This part is devoted to deeply study social networks as representation of online communities. It focuses on four main aspects: data extraction, privacy, security and trust. In particular, Chapter 2 deals with the crucial aspect of extracting data from a multi-social network scenario. The output of this research is a model to generalize concepts, actions and relationships of existing social networks.This can be used to produce meta-APIs for data extraction. The system implementing this model relies on technologies such as FOAF, XFN and the APIs provided by the OSN. This system is used throughout the whole thesis as a mean for dealing with the of multi-social network paradigm described above.

Concerning the studies on privacy issues, Chapter 3 aims at analyzing two key aspects: *(i)* the behavior of users belonging to both Facebook and Twitter especially focusing on privacy and disclosure of personal information, friendship and user activity; *(ii)* user perception of privacy and the robustness of Facebook privacy settings. This chapter ends with the presentation of a possible attack to the mechanism adopted by Facebook to keep users' friend list private.

The investigation related to the field of security is presented in Chapter 4. In particular, it focuses on security issues for social network mobile apps. Nowadays, mobile applications security is one of the most important topics in the field of information security, due to their pervasiveness in people's life. Among mobile applications, those that interact with social network profiles, have a great potential for development, as they intercept another powerful asset of today cyberspace. Hence, in this scenario the lack of fine-grained control when an application uses the APIs of a social network to access a profile may lead to security threats. To overcome this problem, Chapter 4 describes an effective solution to these issues for Android applications.

Finally, Chapter 5 discusses the topics of trust inside online communities. Many real-life reputation models show several drawbacks making the systems adopting them vulnerable to users' misbehavior. Therefore, this Chapter proposes a reputation model abstractly considering service providers, users and feedbacks, and implementing the theoretical notion of certified reputation to concretely define a strategy to *normalize* feedback scores towards reliable values. This model is then applied to the case of TripAdvisor, by proposing a solution to improve its dependability neither increasing invasiveness nor reducing usability of the system.

2

Online communities: Social Networks

Online Social Networks have become so pervasive in people's lives that they can play a crucial role in the design and the development process of applications. Moreover, from a research point of view, OSNs can be considered as reference models for the analysis of online communities. At this moment, standard networking programming is not adequate to support social-network-based programming in large, according to software engineering principles of genericity and polymorphism. This drawback is evident when applications should be built on top of multiple social networks still keeping a user-centered vision. Indeed, the heterogeneity of social networks does not allow the development of software with a suitable abstraction level. In this chapter, we cover the above gap by defining and implementing a model aimed at generalizing concepts, actions and relationships of existing social networks. We will leverage this model for all the approaches based on the multi-social network paradigm described above.

2.1 Background

This section provides the background necessary to fully understand the concepts presented in this chapter. First, it discusses the main features that differentiate a social network from a regular website, then it lists the social networks we analyze to build our model and, finally, it describes the reference scenario of this chapter, which involves social networks altogether.

Online social networks (OSNs) provide powerful technical features to make communication among users easy. Their backbone consists of public profiles, which collect personal information and interests, and an articulated list of friends who are other users of the system. When a user joins a social network, usually he has to fill his own profile with descriptors, such as age, location, interests, photos and multimedia contents. Moreover, an OSN models entities and connections among them. Entities are often individuals who connected to each other by personal relationships, interactions, or information flows. The collection of friends is not simply a list of close profiles. It represents a microcosm inside the social network, where each user can interact with others. Because a friend list is visible to everyone, users can trace friend links. A new participant can find and add a new friend using the friend lists of the other users.

Profiles and friend lists are only two key features of social networks. The third feature allows users to write comments, which are prominently displayed and are visible to anyone accesses the profile of the user who generates it.

The three features (profiles, friends lists and comments) represent the basic structure of a social network. Moreover, all social networks can have a set of basic functionalities which are considered essential to qualify them as a social networking service. These functionalities are:

- the ability to set up and customize a personal profile by simple forms;
- an utility that allows members to reference other users in their posts;
- a feature allowing users to make a granular control of shared information (privacy settings);
- the ability to block an unwanted member in order to exclude him from the friend list;
- a homepage containing personal information, notes and individual picture albums.

Most of the OSNs include also many other proprietary functionalities, such as instantaneous messages, photo tagging tools, notifications, photo and video sharing, the ability to own, form or be member of a group or a community within the network, and to include new "social applications" or gadgets.

In our article, we focus on some specific social sites chosen according to their popularity and specificities. However, most of the other social networks not mentioned here, have functionalities similar to that described below. The social network data are referred to [222].

Twitter is a microblogging and an online social networking service that allows users to exchange short (140-character) messages called *tweets*. We choose it because it currently ranks as one of the leading social networks worldwide based on active users. As of the fourth quarter of 2014, Twitter has 288 million monthly active users. The peculiarity of Twitter lies in its efficiency to spread out information instantaneously: it allows one person to inform millions of people in seconds, and suddenly to see responses and direct replies.

Facebook is the biggest social network in the world and allows people to connect to each other, upload an unlimited number of photos, post links and videos. At the end of 2014, it has more than 1.39 billion global monthly active users. One of the winning factor of `Facebook` is its user centric vision, as it is the first social network to literally focus all of the attention on the user and what he wants to express and portray about himself.

`LinkedIn` is a business-oriented social networking service. With close to 347 million members worldwide in December 2014, it is one of the most popular social network in terms of active users and the most trustworthy source of professional content according to UK business professionals. Available in more than 200 countries, its website focuses on business connections and industry contacts for employers and working professionals, allowing companies to present themselves and users to find job listings, to build their career and to stay in touch with their connections in their area of expertise.

`Flickr` is an image hosting and video hosting website and an online community and a multimedia networks. The main aim of Flickr is to allow users to upload their photos as well as organize and share them with other users. We choose it because it is one of the most used online photo management and sharing application in the world. It provides users with a massive online photo storage allotment of a whole terabyte.

`Google+` is a social networking and identity service owned and operated by Google. Its 359 millions of active users make it a leading social networks worldwide. It is intended to integrate all Google services (Gmail, Google Maps, search, Google Calendar, etc.) into one cohesive network, incorporating everything that searchers use at Google into a comprehensive social and content dashboard.

`LiveJournal` is a community publishing platform and a social networking service where users can keep a blog, journal or diary. It has more than 50 million journals on different topics like politics, entertainment, fashion, literature and design. We choose `LiveJournal` because it has been well studied by social network analyzers in the past.

`Advogato` is an online community and social networking site dedicated to the development of free software. It represents a resource for free software developers because it provides a research testbed for group trust metrics and other social networking technologies. This site is mentioned for its early adoption of the FOAF ontology as an alternative method for showing user information.

`about.me` is a personal web hosting service that ties together users of other social-networking sites. It also includes analytics that let users track things like how many people viewed their `about.me` page and which other social-networking profiles they viewed from there. We choose it because of its main feature of linking together in the user profile relevant external sites and multiple social networking websites such as `Facebook`, `Flickr`, `Google+`, `LinkedIn`, `Twitter`, `Tumblr` and `YouTube`.

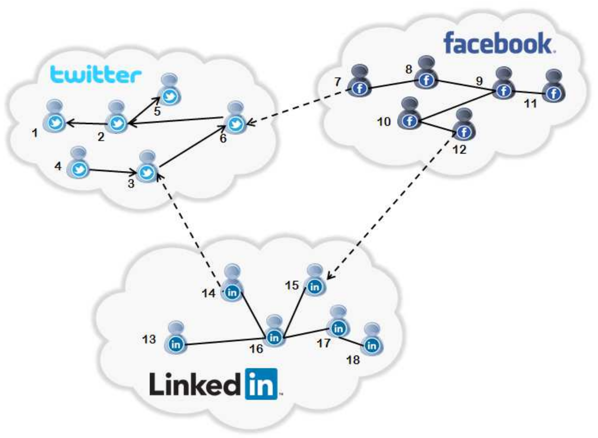


Fig. 2.1: An example of a multiple social network scenario.

Social networks altogether form a more complex scenario in which users interaction assumes a relevant importance. This interaction is enabled by the presence of users who have multiple profiles in different social networks and adopt particular kind of edges, called me edges, to link them [63]. Figure 2.1 shows a graphical representation of a possible scenario involving three social networks, namely Twitter, Facebook, and LinkedIn. In this example, we have that nodes from 1 to 6 are Twitter accounts, nodes from 7 to 12 are Facebook accounts and, finally, nodes from 13 to 18 are LinkedIn accounts. As for edges, they represent friendship relationships among users. However, while edges among Facebook and LinkedIn actors are bidirectional, those among Twitter users are directed, according to the typology of relationship allowed by the social network. Finally, edges $(14, 3)$, $(12, 15)$ and $(7, 6)$ represent me edges and connect accounts of the same user on different OSN.

2.2 Design specification

In Section 2.1, we focused on the general services provided by the most popular OSNs. Their study is one of the targets of our work. As it can be recognized by analyzing the technical details described in the sequel of the section, there is strong heterogeneity in the representation of concepts among different social networks. For instance, contacts are represented by *friends* in Facebook and the relationship is symmetric, while they are represented by *followers* and *followings* in Twitter and the corresponding relationship is not symmetric. Again, the concept of appreciation becomes *+1* in Google+ and *endorsement* in about.me. Importantly, similar concepts can mapped to each other but they have in general different features. Thus, an integration step is necessary for our purpose. In this section, we prepare this integration step by grouping the main technical entities into a number of categories to which the formal model presented in the next section maps. In particular, we aim at modeling the following entities.

2.2.1 Profile

Social network sites are built around user profiles, a form of individual (or group) homepage, which provides a description of each registered user. Profiles are constructed by filling out forms on the site.

As for `Twitter`, at the moment of registration, a user can create his profile typing his name, username, password and email address in the registration form. People often use their real name without the spaces as username. After typing in the CAPTCHA words from the image, a user can create his account. When a user is logged in, he can upload a profile picture and start following other people. Moreover he can complete his profile adding a short biography, a position (the place where he lives) and a link to his website or to one of his account on other social networks.

Similarly, if a user wants to sign up for `Facebook` he has to enter in the suitable page his full name, a valid email address, a password, his gender and birthday. As a second step, the user can complete his timeline, which is his personal profile. Timeline includes everything from uploading a profile picture and cover photo to outlining user employment history, determining his relationship status, declaring web link toward the profile of the same user in other social networks.

To join `LinkedIn`, a user has to fill out his biography with information, such as past and present employment, education, skills and web links. It is also possible to add a user profile photo. The "Headline" and the "Professional Summary" section of the profile are useful to highlight user experiences.

Whereas if a user wants to register a `Flickr` account, first he has to create a Yahoo! Mail account. Then the user has to choose his `Flickr` screen name that will be the user name for the site. After the first access, a user can fill out his profile adding some personal information as gender, birthday, web link, occupation, hometown, relationship status, interests.

As for `Google+`, a user with a gmail account can automatically sign in, otherwise, he has to create it filling in his name, preferred username, password, birthday, gender, mobile Phone and other email address. Once the user logs in, he can choose his username, upload a picture and complete further information about himself, such as where he went to school or where he works.

A user can sign in `LiveJournal` platform via one of his account (`Facebook`, `Google+`, `Twitter`, etc.). After this step, he has to complete his profile adding a profile picture, username, gender, birthday, education, web links, interests, biography and position.

Advogato allows users to create a profile page and a blog. At the moment of the registration, the user has to provide a valid email address, a username and a password. The rest of information (such as name, surname, notes) is optional, but is useful in order to be certified by other Advogato users.

Finally, about.me is characterized by its one-page user profiles, each with a large background image and short biography. At the moment of registration a user has to fill the suitable form with his username, email, password for the site and at a second step short biography, a short description, a profile image and a background image.

2.2.2 Links to external social networks

An important feature provided by all the social networks considered in this chapter is the possibility for a user to add in his profile a link toward one of his accounts in another social site or external website. This feature is typically enabled during the creation of the user profile. It is of particular interest because it encodes the basic information allowing the possibility of seeing different social sites as members of a Multiple-Social-Network environment.

2.2.3 Friendship

After creating a profile, participants are asked to invite their friends to the site or to look at others' profiles and add those people to their list of friends.

In Twitter, a user can follow another user, becoming his *follower*. Only if this user follows him back the relationship is bidirectional.

Differently from Twitter, Facebook requires approval for two people to be linked as friends. When someone links another as a friend, the recipient receives a message asking for confirmation. Indeed, Facebook friendship is bidirectional, hence, once a user accepts a friendship request of another user they become mutual friends.

LinkedIn allows registered users to maintain a list of contact details of people with whom they have some level of work relationship, called *connections*. When a user establishes a relationship with another user, he declares a sort of mutual friendship and, from this moment on, he will see all the updates of this new connection in his homepage.

Flickr follows a strategy similar to that of Twitter about contacts. These social ties may or may not be reciprocated. Only users who include each other as contact have a reciprocal relationship. Once a user adds another user as contact, he can further distinguish the relationship with this user by labeling him as friend and/ or member of his family or just keeping him only as a following contact.

Among the functionalities of Google+ there is the possibility of adding users in a friend list. In particular, a user can assign his contacts to one or more “circles” (such as friends, colleagues or acquaintances), which is a way of categorizing and organizing people.

In LiveJournal two users can list each other as friends mutually, or one of them can *follow* the other without reciprocation, like in Facebook.

Finally, neither Advogato nor about.me provide the user with the possibility to create any list of friends.

2.2.4 Resources

A Social network resource is a Web asset such as a status update, a photo, a web link or a video created and loaded by a user in his profile.

Twitter resources can be shared only inside a *tweet*. They can be photos, videos, web links or comments. Once a user generates a tweet, it is publicly posted on his Twitter profile. Moreover, each *tweet* can be associated with one or more hashtags describing it, and/or some references to other user profiles. The stream of tweets of a user is called *timeline*.

Concerning Facebook, a user can publish one or more resources in a *post* in his timeline. The post can contain photos, videos, comments, resources coming from other social platforms, hashtags and references to other users. A user can also add in his “diary” (his Facebook profile) notes, photos or videos without publishing them in a post. Photos and videos can be uploaded in specific albums.

As for LinkedIn, a user can add a resource like a new item or a new file in his profile. He can also embed a comment, a photo, a web link or a video in a new status update. Also skills representing specific technical expertise can be seen as a typology of resource, which are posted by users to describe their ability. This way, his *connections* can like it, comment it and share it on their “wall”.

In Flickr, when a user upload an image or a video he can optionally add a text description to the resource. The images a Flickr photographer uploads are stored into his sequential “photostream”, which is the basis of a Flickr account.

As far as Google+ is concerned, a user can share messages, links, photos or videos with everyone or only with those within designated circles, but he can also create his own photo albums and add his photos or videos inside them.

In LiveJournal, examples of resources are images, videos and audio files. Resources can be uploaded by users in their blog post.

In Advogato, once a user account has been certified by other trusted users, he will be able to post to the news flow, create projects, or syndicate his blog to the Advogato recentlog from his existing blogging site. The only form of resource available in Advogato is the *article*.

Finally, in about.me there is not an explicit concept of resource. Indeed comments and appreciations are allowed directly on the user profile and users cannot upload any photos or videos, except for the profile and background picture.

2.2.5 Actions on resources

So far, we stated that in addition to the content that members add when they create their own profiles, social network sites typically provide the possibility to share resources. After a resource is published by a user, several actions can be performed on this resource: other users can appreciate it, or re-share it, or it can be associated with a user through a mention on his profile.

Hereafter, we list the main possible actions a user can do on a resource according to the different social networks analyzed in this chapter.

Once a user write a *tweet* in Twitter, it will appear on the homepage of all his followers, who can *reply* to it, make it one of their *favourites* or *retweet* it (that is, forwarding it again on their own timeline). A *tweet* can contain also a user *mention*. It can be done using the symbol @ followed by the referenced username. To categorize *tweets* by keyword, people use the hashtag symbol # before a relevant keyword or phrase (no spaces) in their *tweets*. Hashtags are indexed to make it easier to find conversations about that topic.

As for Facebook, when a resource is posted, a user can comment it and/or give a positive feedback through the *like* button. Users can *like* all types of resources, such as: status updates, comments, photos, other user profiles, links posted by their friends and adverts by clicking the *like* button at the bottom of the content. This makes the content appear in their friends' "News Feeds". Moreover, users who are interested in or agree to a post, can share it again in their timeline (*re-post*). This allows a very fast propagation of posts inside Facebook.

The concept of referencing users in status updates has been introduced as an attempt to imitate Twitter. This means putting the name of a user, a brand, an event or a group in a post in such a way that it is linked to the wall of the Facebook page being tagged. Thus, the post appears in news feeds for that page, as well as those of selected friends. This is done by using the @ symbol followed by a person's name.

The same symbol allows users to tag people in photos or videos taken of them. This functionality is a peculiar feature of Facebook. Whereas using the # symbol followed by a tag word in a status allows a user to create a hashtag. As explained before, this metadata tag allows grouping of similarly tagged messages and support the search for messages referring to a specific topic represented by that hashtag. Every hashtag on Facebook has its own unique URL and this allows to search for specific topics from the Facebook search bar.

Clicking on *like* option on LinkedIn presents some differences w.r.t. the Facebook *like* function. Indeed, on LinkedIn, when users click on the *like* link underneath the various updates, this immediately forwards that particular update out to all of the user first level *connections*. The *share* option, instead, allows users to either redistribute the article (and partially modify it) as an update to their *connections*, post it to a group (or multiple groups), or forward it in a private message. The *comment* link allows users to comment on someone's update.

Similarly to what happens in Twitter, also in LinkedIn while a user publishes a resource he can mention one of his *connections* with the @ symbol. He can also use a keyword as hashtag using the # symbol.

Furthermore, companies can post information about themselves, list jobs and search for potential candidates. Finally, LinkedIn allows users to endorse each other's skills.

As for Flickr, by clicking on a photostream image, it is possible to open it in the interactive *photopage*, thus allowing users to comment it and to embed it on external websites. Moreover, images can be added to a user favourite list or to user galleries.

Users may label their uploaded images with titles and descriptions, and images may be tagged either by the uploader or by other users, if the uploader permits it.

The main Google+ page consists of a "stream" of updates, conversations and shared content. A user can make comments underneath content shared by other users, and he can appreciate contents clicking "+1" on it. A user can also re-share contents within his circles. Google+ provides the referencing functionality in its posts. A user can mention another user using the + or @ signs.

Moreover, a user can insert some hashtag in his comments similarly to what happens for Twitter. The main differences with the hashtag of Twitter is that here the system automatically adds hashtags (recognizable by different colors), too.

As for LiveJournal, users can interact with resources in different ways. For instance, a user can leave a comment on a post of another user or share it in his blog. He can also add to "Memories" a post. The Memories feature on LiveJournal allows the organization of favorite resources with a keyword-based archive system. Thanks to this functionality, a user can also add tags, or descriptive keywords, to his own resources.

As far as about.me is concerned, an interesting characteristic is the possibility to make *compliments*, which is a form of *like* made on user pages. There are different kinds of compliments a user can do and about.me let users choose among them, whether it was a professional compliment or a more personal one. The company provides also a service called *collections*: a user can organize various profiles into public or private *collections*. In order to bring *collections* and *compliments* together, `about.me` introduced the Dashboard, which included "PeopleFeed". This let users see activities on their page, including views, who visited and complimented their page. *Replies*, instead, let users respond to activity on their page.

All the features of the OSNs described in this section are mapped by our model, which is formalized in the next section.

2.3 The conceptual model

In the previous section, we have identified eleven technical entities, of which three concepts and eight relationships. Now, we want to formalize the so described environment into an abstract multiple-social-network model. To do this, we adopt a direct graph $G = \langle N, E \rangle$, in which nodes represent the concepts and edges encode the relationships. Therefore, the set of nodes is partitioned into three disjoint sets P, R, and B, which correspond to the set of social profiles, the set of resources, and the set of bundles (which are resource containers), respectively. Further, the set of edges is partitioned into eight disjoint sets F, M, Pu, S, T, Re, L, and Co, each corresponding to one of the eight relationships identified in Section 2.2.

Let us start with the description of nodes. An element of P models the *profile* of a user on a social network and consists in the tuple ⟨`url, socialNetwork, screen-name, [personalInformation], [picture]`⟩. In this tuple `url` is the Web address that identifies and localizes the profile, and `socialNetwork` is the commercial name of the social network which the profile belongs to, `screen-name` is the name chosen by the user who registered the profile to appear in the home-page of the profile or when posting a resource, and, finally, `personalInformation` and `picture` are the information and the image which the user inserted as related to the profile. The two last elements of the tuple are optional (i.e., they can be null).

The set R models *resources* of the Web or created by users. A resource is represented by a tuple ⟨`url, type, [description], [date]`⟩, where `url` is the Web address to access the resource, `type` indicates the type of the resource content, and finally, `description` and `date`, which are optional, represent the string, inserted by the who published the resource, describing the resource itself and the publishing date, respectively. For example, the most viewed video on YouTube is a resource represented as ⟨`'https://www.youtube.com/watch?v=9bZkp7q19f0', 'video/mp4', 'PSY - GANGNAM STYLE', '07/15/2012'`⟩.

Our model includes the *bundle* set B. Indeed, commonly users do not handle a single resource, but most of the actions they do (e.g., publishing or sharing) involve more resources simultaneously. For example, a user can publish more photos or videos, can include a comment, and so on. In our model, we include all resources handled simultaneously by a user in a bundle. A bundle is represented by a tuple $\langle$`uri, [description], [date]`$\rangle$, where `uri` is the identifier of the bundle, `description`, which is optional, is the string chosen by the user to be shown with those resources and, finally, `date` represents the publishing date. As we will see next, we represent the inclusion of a resource into a bundle by means of *containing* edges.

Relationships among profiles, resources and bundles are represented by direct edges of a graph. As already stated earlier, the set E of edges is partitioned into eight disjoint sets, named F, M, Pu, S, T, Re, L, and Co, each corresponding to one of the eight relationships identified in Section 2.2.

The *follow* edge set $F \subseteq E = \{p_s, p_t \mid p_s, p_t \in P\}$ models the fact that in the (source) profile p_s, it has been declared a certain type of relationship towards the (target) profile p_t. This kind of edge models different relationships. For example, on `Facebook` or `Flickr`, it models friendships, on `LinkedIn`, job contacts, and, on `Twitter`, *followers*. Observe that, typically, this kind of relationship occurs between users of the same social network, because it is presumable that a social network does not have interest in promoting links to profiles of another (competitor) social network.

The *me* edge set $M \subseteq E = \{p_s, p_t \mid p_s, p_t \in P\}$ denotes that the user with profile p_s has declared in this profile to have a second profile p_t. This edge allows a user to provide a link to its profile (typically) on a different social network or (sometimes) on the same social network (as a sort of alias).

The *publishing* edge set $Pu \subseteq E = \{p_s, b_t \mid p_s \in P, b_t \in B\}$ indicates that the user with profile p_s has published in this profile a bundle b_t. This edge models one of the typical actions a user does when enriches his/her profile by publishing resources.

The *shared* edge set $S \subseteq E = \{b_s, b_t \mid b_s, b_t \in B\}$ specifies that the bundle b_s (published by a user) is derived from an already published bundle b_t. This type of edge is used when a user shares an existing bundle. Indeed, this action is represented by two edges: a publishing edge (as described before) and a shared edge from the new bundle to the existing one.

The *tagging* edge set $T \subseteq E = \{p_s, br_t, w \mid p_s \in P, br_t \in B \cup R$ and w is a word$\}$, denotes that the user with profile p_s assigned the word w to describe a bundle or a resource br. By means of the tag mechanism, users contribute to resource labelling, which is necessary to carry out several actions on resources, such as searching or classification.

The *referencing* edge set $Re \subseteq E = \{b_s, p_t \mid b_s \in B, p_t \in P\}$ models the fact that a bundle b_s includes a reference to the profile p_t. For example, this occurs when a *tweet* includes a user account name.

The *like* edge set $L \subseteq E = \{p_s, pbr_t \mid p_s \in P, pbr_t \in B \cup R \cup P\}$ describes the information that a user with the profile p_s expressed a preference/appreciation for a bundle, a resource or another user profile pbr_t.

The *containing* edge set $Co \subseteq E = \{b_s, r_t \mid b_s \in B, r_t \in R\}$ indicates that a bundle b_s contains the resource r_t. For example, when a user publishes a photo p and includes a comment c, this action is modeled by creating a bundle b with a description c, a resource p, and finally, a *containing* edge from b to p.

The model defined above is able to represent data coming from multiple social networks. As a consequence, this model may appears more complex than those typically adopted in Social Network Analysis. However, it is worth noting that some measures used in this field can be still calculated in a easy way by suitably *pruning* the graph $G = \langle N, E \rangle$ underlying the model. For example, the friendship degree distribution of social network users has to be computed on the graph $G' = \langle N', E' \rangle$, in which $N' = P$ and $E' = F$, that is, the subgraph obtained from G by maintaining only Profile nodes and Follow edges. Analogously, other topological features, such as clustering coefficient and assortative coefficient, are computed also on the same subgraph G'.

After defining the conceptual model, we will show how to practically map real-life data from social networks to each component of the model, in such a way to build a data structure that can be used at application level (as we will show in Section 2.5).

2.4 Building the model

Information necessary to build the model can be extracted from social networks via four technologies: (i) APIs provided by the social network; (ii) FOAF datasets; (iii) XFN microformat; and (iv) HTML parsing.

As for the first technology, social network APIs are a platform available for developers which allow the access to social-networks data so as to create applications on top of them. Usually, there are different kinds of APIs each providing specific services. Among them, the most commons are the REST API, the Search API and the Streaming API. Specifically, the REST APIs allow operations such as insert, update or deletion to be performed. The Search APIs, instead, are useful to query the database and, finally, the Streaming APIs are conceived for applications that need to receive real-time updates (such as, new posts or feeds).

```
{
  "id": "1587099156",
  "first_name": "Serena",
  "gender": "female",
  "last_name": "Nicolazzo",
  "locale": "en_GB",
  "name": "Serena␣Nicolazzo",
  "username": "serena.nicolazzo"
}
```

Fig. 2.2: An example of the output of the `Facebook` Graph API

The second possible strategy to extract information from social network relies on FOAF datasets. The FOAF project focuses on the creation of a machine-readable ontology describing friendship relationships among users. FOAF data sources allow the representation of a whole social network without the need of a centralized database. As a matter of fact, by relying on this technology, it is possible to represent the information concerning a user account, along with the corresponding contacts and activities, through an RDF graph serialized as an XML document, according to the W3C RDF/XML syntax.

The third option makes use of XFN microformat. It allows for the representation of the kind of relationship existing between two user accounts. This is obtained by empowering the set of values that the `rel` attribute of the HTML tag <a> (which represents a link) can assume. In our case, we focus on the value "me" (``rel=`me'``) which indicates that the corresponding link represents a `me` edge.

The last data extraction strategy leverages on HMTL parsing. Processing HTML to obtain social data is the most intricate procedure. Parsing requires much time because it needs to analyze all context information from the page source code. It is a low-level way of dealing with social data. Because the code written depends on the HTML page structure, it is not stable (due to the frequent graphical changes). For this reason, this strategy needs continue maintenances. However, it remains a valid alternative when other more practical solutions (like APIs, for instance) are not available.

Now we will show some significant examples on how the information represented by our model are extracted from social networks.

As for the user profile P described in Section 2.3, we recall that it consists in a tuple ⟨url, socialNetwork, screen-name, personalInformation, picture⟩. For example, to extract the information to build the profile of a `Facebook` user, we use the `Graph APIs`, accessible through the url `http://graph.facebook.com/{user-id}` or `http://graph.facebook.com/{screen-name}`. The output of this API is a JSON file (see, for instance, Figure 2.2).

```
1  <?xml version='1.0'?>
2  <rdf:RDF
3    xml:lang="en"
4    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
5    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
6    xmlns:foaf="http://xmlns.com/foaf/0.1/"
7    xmlns:ya="http://blogs.yandex.ru/schema/foaf/"
8    xmlns:lj="http://www.livejournal.org/rss/lj/1.0/"
9    xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#"
10   xmlns:dc="http://purl.org/dc/elements/1.1/">
11   <foaf:Person>
12     <foaf:nick>antoninonocera</foaf:nick>
13     <foaf:name>real_name</foaf:name>
14     <foaf:openid rdf:resource="http://antoninonocera.livejournal.com/"/>
15     <foaf:weblog rdf:resource="http://antoninonocera.livejournal.com/"/>
16     <foaf:homepage rdf:resource="*****" dc:title=""/>
17     ...
18     <foaf:knows>
19       <foaf:Person>
20         <foaf:nick>contact1</foaf:nick>
21         <foaf:member_name>contact_real_name</foaf:member_name>
22         <foaf:tagLine></foaf:tagLine>
23         <foaf:image>http://l-userpic.livejournal.com/****/****</foaf:image>
24         <rdfs:seeAlso rdf:resource="http://contact1.livejournal.com/data/foaf"/>
25         <foaf:weblog rdf:resource="http://contact1.livejournal.com/"/>
26       </foaf:Person>
27     </foaf:knows>
28       ...
29   </foaf:Person>
30 </rdf:RDF>
```

Fig. 2.3: An XML-serialized FOAF document

We can extract from this JSON the user id, his username (which correspond to our notion of screen-name) and his personal information like: first name, last name, gender, locale (chosen language). The field *url* available in our social profile object can be obtained as `http://www.facebook.com/{screen-name}`, whereas the field *picture* can be obtained by another call to the `Graph APIs`, specifically by accessing the url `http://graph.facebook.com/{user-id}/picture`.

Many social networks are equipped also with FOAF datasets. As an example, we show how *follow* edges can be obtained for the social networks `LiveJournal` and `Advogato`. The FOAF datasets for both social networks are reachable through the specific URLs `http://{screen-name}.livejournal.com/data/foaf` (for `LiveJournal`) and `http://www.advogato.org/person/{screen-name}/foaf.rdf` (for `Advogato`). An example of an XML serialization of a FOAF document is shown in Figure 2.3. In this document, the information needed to build an edge of the set *follow* can be extracted from lines 11 to 29. Specifically, the element `<foaf:Person>` indicates the beginning of the portion of the document where information about a user, his contacts and, often, his activities are reported. The information about each contact is encoded as a `<foaf:Person>` nested inside a tag `<foaf:knows>`.

Concerning the information about *me* edges, it can often be extracted through the XFN microformat. Some examples of social networks adopting this standard to rep-

```
1 <a class="OLa␣url␣Xvc" href="http://www.youtube.com/channel/UCIUcwh3yFufPSnCbyrUyTBQ"
2 rel="me" target="_blank" title="UCIUcwh3yFufPSnCbyrUyTBQ">YouTube Channel of Antonino Nocera</a>
```

Fig. 2.4: An example of a *me* edge using XFN.

```
1  <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2  <job>
3    <id>1511685</id>
4    <expiration-timestamp>1304030488000</expiration-timestamp>
5    <company>
6      <id>229433</id>
7      <name>Cloudera</name>
8    </company>
9    <position>
10     <title>Technical Writer</title>
11     <location>
12       <name>San Francisco Bay Area</name>
13       <country>
14        <code>us</code>
15       </country>
16     </location>
17   </position>
18   <location-description>San Francisco or Palo Alto,CA</location-description>
19   <job-poster>
20     <id>hQ4ruu3J2q</id>
21     <first-name>Paul</first-name>
22     <last-name>Battaglia</last-name>
23     <headline>Technical Writer at Cloudera</headline>
24   </job-poster>
25 </job>
```

Fig. 2.5: An example of the output of the `LinkedIn` Job Lookup API.

resent *me* edges are `about.me`, `Advogato`, `Facebook`, `Flickr`, `Google+`, and `Twitter`. Figure 2.4 shows the code representing a *me* edge in the social network `Google+`. The code at line 2 represents the explicit declaration that the corresponding link encodes a relationship of type *me*.

Another interesting example regards the extraction of the information needed to build a *publishing* edge. Consider the social network `LinkedIn`. It provides a search API, called Job Lookup API, to obtain information about jobs that can be accessed at the address `api.linkedin.com/v1/jobs/{job_id}:(id,company,posting-date)`. The XML output produced by the call to this API is reported in Figure 2.5. In this case, when a company proposes a new job position (publishing), we model this event by adding two objects: (i) a bundle and (ii) a *publishing* edge between the profile of the company and the bundle just created (see Section 2.3). As for the bundle, the field `uri` is mapped to the element `<id>` (line 3), whereas the field `description` is obtained from the elements `<company>` (lines 5-8), `<position>` (lines 9-17), and `<localition-description>` (line 18). The *publishing* edge is associated with the user profile whose identifier is specified by the element `<job-poster>` (lines 19-24) and has the new created bundle as target.

Now, consider the case of the publishing of a new tweet containing a resource and referencing another user in the `Twitter` social network. Our model represents this action by adding the following objects: (i) a bundle, (ii) a resource, (iii) a *publishing* edge, (iv) a *containing* edge, and (v) a *referencing* edge. In this case, all information required by our model is extracted from `Twitter` by means of the method *GET statuses/user_timeline* of the `Twitter` APIs. Figure 2.6 shows an example of the output of this API. Specifically, line 6 is the tweet identifier and is mapped to the field `uri` of the bundle. The bundle field `description` is obtained by suitably parsing lines 24-30. The bundle is linked to the publisher user by means of a *publishing* edge. As mentioned above, a new resource is added and associated with the bundle by means of a *containing* edge. Information needed to create the resource is extracted from lines 11-23. In particular, the field `url` is obtained from line 15, the field `type` from line 20, and, finally, the field `description` is extracted from lines 19, 20 and 21. The tweet in the example references another user and this action is modeled by adding a *referencing* from the bundle to the user specified on lines 9 and 10.

An important feature, common to almost all social networks, is the possibility to appreciate a resource or another user profile. In our model, this concept is represented by means of the *like* edge. Consider the social network `about.me`, in which a user is allowed to favor another user profile, thus making an "endorsement". Information about this action is obtained by calling the method `http://api.about.me/api/v2/json/user/view/` of the `about.me` API followed by the desired user *screen_name*. Figure 2.7 reports an example of the output of this method called on the profile of the author of this thesis. The returned information has to be seen from the "caller" point of view (i.e., the authenticated user), therefore the line 21 indicates that the user *snicolazzo* is in the favourite list of the user calling this API method. According to our model, a *like* edge from the authenticated user to the user with the given *snicolazzo* is created. A similar reasoning can be applied also for "g+1" of `Google+` and "Like" of `Facebook`.

So far, we have seen how to extract information from different social networks and how to map them to the concepts defined in our model. Once this mapping has been done, a data-structure is obtained. It can be serialized using the XML language. In the following, we will show some details about the XML schema designed for our model.

Figures 2.8 shows the mind map of this XML schema [30]. The root element is *SocialGraph* and contains two unbounded sets of elements, namely *SocialNode* and *SocialEdge*. An element *SocialNode* is specialized in one of the following complex types: *SocialProfile*, *Resource*, or *Bundle*. The element *SocialEdge* is specialized in one of the following complex types: *Follow*, *Me*, *Publishing*, *Tagging*, *Shared*, or *Referenc-*

```
{
"coordinates": null,
"favorited": false,
"truncated": false,
"created_at": "Wed␣Aug␣29␣17:12:58␣+0000␣2012",
"id_str": "240859602684612608",
"entities": {
  "hashtags": [ ],
  "user_mentions":[{"indices": [3, 10],"id_str": "<user_id>","screen_name":
  "<user_screen_name>","name": "<user_real_name>", "id": <user_id>}]
  "media": [{
    "id": 266031293949698048,
    "id_str": "266031293949698048",
    "indices": [17, 37],
    "media_url": "http://pbs.twimg.com/media/A7EiDWcCYAAZT1D.jpg",
    "media_url_https": "https://pbs.twimg.com/media/A7EiDWcCYAAZT1D.jpg",
    "url": "http://t.co/bAJE6Vom",
    "display_url": "pic.twitter.com/bAJE6Vom",
    "expanded_url":"http://twitter.com/BarackObama/status/266031293945503744/photo/1",
    "type": "photo",
    "sizes": {...}
  }]
},
"in_reply_to_user_id_str": null,
"contributors": null,
"text": "Introducing␣the␣Twitter␣Certified␣Products␣Program:␣https://t.co/MjJ8xAnT",
"retweet_count": 121,
"in_reply_to_status_id_str": null,
"id": 240859602684612608,
"geo": null,
"retweeted": false,
"possibly_sensitive": false,
"in_reply_to_user_id": null,
"place": null,
"user": {...},
"in_reply_to_screen_name": null,
"source": "<a␣href=\"http://sites.google.com/site/yorufukurou/\"␣rel=\"nofollow\">YoruFukurou</a>",
"in_reply_to_status_id": null
}
```

Fig. 2.6: An example of the output of the API method *user_timeline*.

ing. Each complex-type in this XML Schema is defined according to the corresponding objects defined in Section 2.3. Figure 2.9 reports a fragment of the XML schema allowing the serialization of our model. The complete XML Schema is available at `http://www.infolab.unirc.it/OSNmodel.html`.

We conclude this section by showing in Figure 2.10 an example of a fragment of an XML document derived from the XML Schema described above.

```
{
"status": 200,
"profile": "http://about.me/snicolazzo",
"user_name": "test␣account",
"first_name": "test22",
"last_name": "tester",
"display_name": "test22␣tester",
"header": "my␣headline",
"bio": "test␣this␣is␣one!!!!",
"background": "http://about.me/.../snicolazzo_1326415784_79.jpg",
"mobile_background": "",
   "email_searchable": true,
   "email_public": false,
   "avatar": "http://about.me/.../snicolazzo_1325746595_83.jpg",
   "img_base_url": "http://about.me/.../thumbnail",
   "thumbnail_291x187": "http://about.me/.../291x187/snicolazzo.jpg",
   "thumbnail1": "http://about.me/.../803x408/snicolazzo.jpg",
   "thumbnail2": "http://about.me/.../260x176/snicolazzo.jpg",
   "thumbnail3": "http://about.me/.../198x134/snicolazzo.jpg",
"thumbnail4": "http://about.me/.../161x109/snicolazzo.jpg",
"is_fav": true
}
```

Fig. 2.7: An example of the output of the API method *view*.

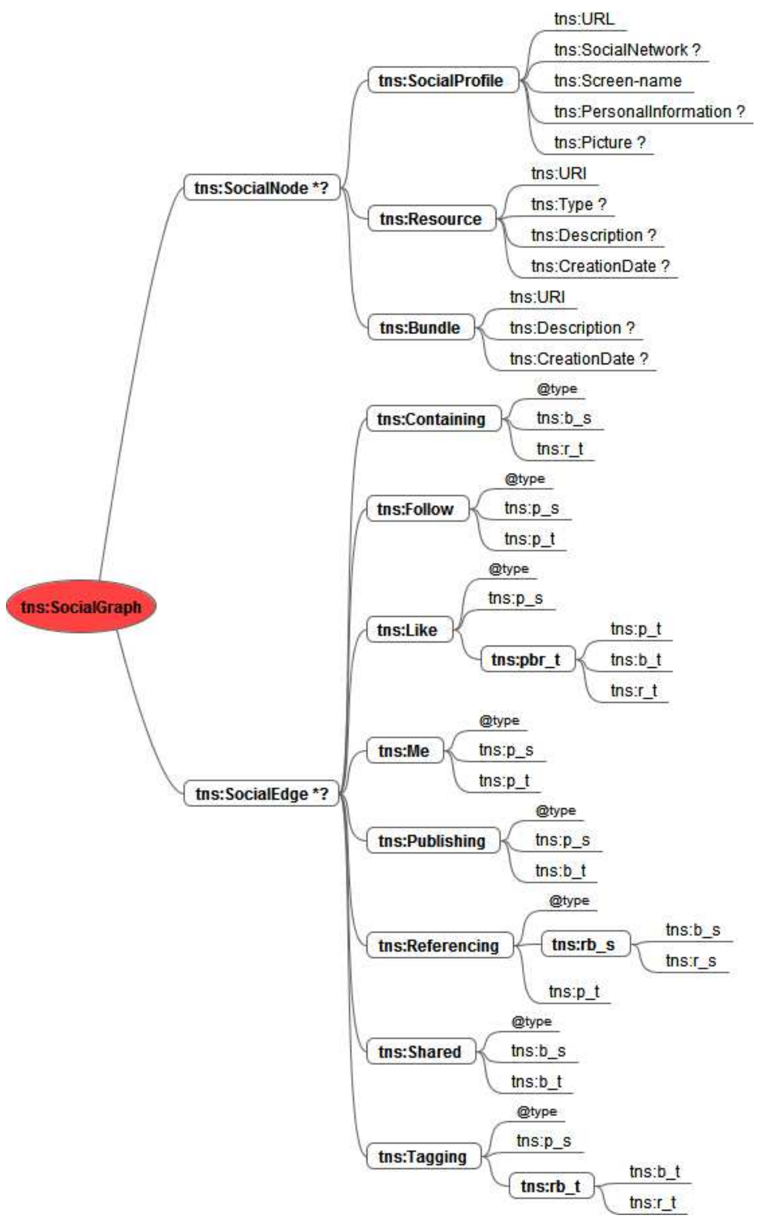


Fig. 2.8: The mind map of our XML Schema.

```
...
<element name="SocialGraph">
 <complexType>
  <sequence>
   <element ref="tns:SocialNode" minOccurs="0" maxOccurs="unbounded" />
   <element ref="tns:SocialEdge" minOccurs="0" maxOccurs="unbounded" />
  </sequence>
 </complexType>
</element>

<element name="SocialNode">
 <complexType>
  <choice>
   <element ref="tns:SocialProfile" />
   <element ref="tns:Resource" />
   <element ref="tns:Bundle" />
  </choice>
 </complexType>
</element>

<element name="SocialProfile">
 <complexType>
  <sequence>
   <element name="URL" type="string" />
   <element name="SocialNetwork" type="string" minOccurs="0" />
   <element name="Screen-name" type="string" />
   <element name="PersonalInformation" type="string" minOccurs="0" />
   <element name="Picture" type="string" minOccurs="0" />
  </sequence>
 </complexType>
 <xs:key name="spkey">
  <xs:selector xpath="SocialGraph/SocialNode/SocialProfile" />
  <xs:field xpath="@URL" />
 </xs:key>
</element>
...
<element name="SocialEdge">
 <complexType>
  <choice>
   <element ref="tns:Containing" />
   <element ref="tns:Follow" />
   <element ref="tns:Like" />
   <element ref="tns:Me" />
   <element ref="tns:Publishing" />
   <element ref="tns:Referencing" />
   <element ref="tns:Shared" />
   <element ref="tns:Tagging" />
  </choice>
 </complexType>
</element>
...
```

Fig. 2.9: A portion of our XML Schema.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<tns:SocialGraph xmlns:tns="http://www.unirc.it/SocialGraph"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.unirc.it/SocialGraph.xsd">

 <tns:SocialNode>
  <tns:SocialProfile>
   <tns:URL>twitter.com/antoninonocera</tns:URL>
   <tns:SocialNetwork>Twitter</tns:SocialNetwork>
   <tns:Screen-name>AntoninoNocera</tns:Screen-name>
  </tns:SocialProfile>
 </tns:SocialNode>

 <tns:SocialNode>
  <tns:SocialProfile>
   <tns:URL>twitter.com/serenanicolazzo</tns:URL>
   <tns:SocialNetwork>Twitter</tns:SocialNetwork>
   <tns:Screen-name>serenanicolazzo</tns:Screen-name>
  </tns:SocialProfile>
 </tns:SocialNode>

 <tns:SocialNode>
  <tns:Bundle>
   <tns:URI>240859602684612608</tns:URI>
   <tns:Description>beautiful photo :-)</tns:Description>
   <tns:CreationDate>2014-12-29</tns:CreationDate>
   </tns:Bundle>
 </tns:SocialNode>

 <tns:SocialNode>
  <tns:Resource>
   <tns:URI>http://pbs.twimg.com/media/A7EZT1D.jpg</tns:URI>
   <tns:Type>photo</tns:Type>
   <tns:Description>A pic of us</tns:Description>
   <tns:CreationDate>2014-12-29</tns:CreationDate>
  </tns:Resource>
 </tns:SocialNode>

 <tns:SocialEdge>
  <tns:Follow type="F">
   <tns:p_s>twitter.com/antoninonocera</tns:p_s>
   <tns:p_t>twitter.com/serenanicolazzo</tns:p_t>
  </tns:Follow>
 </tns:SocialEdge>

 <tns:SocialEdge>
  <tns:Publishing type="Pu">
   <tns:p_s>twitter.com/antoninonocera</tns:p_s>
   <tns:b_t>240859602684612608</tns:b_t>
  </tns:Publishing>
 </tns:SocialEdge>

 <tns:SocialEdge>
  <tns:Containing type="Co">
   <tns:b_s>240859602684612608</tns:b_s>
   <tns:r_t>http://pbs.twimg.com/media/A7EiDT1D.jpg</tns:r_t>
  </tns:Containing>
 </tns:SocialEdge>
 <tns:SocialNode>
  <tns:SocialProfile>
   <tns:URL>www.facebook.com/antonino.nocera.35</tns:URL>
   <tns:SocialNetwork>Facebook</tns:SocialNetwork>
   <tns:Screen-name>Antonino Nocera</tns:Screen-name>
   <tns:PersonalInformation>Male</tns:PersonalInformation>
  </tns:SocialProfile>
 </tns:SocialNode>
 <tns:SocialEdge>
  <tns:Me>
   <tns:p_s>twitter.com/antoninonocera</tns:p_s>
   <tns:p_t>www.facebook.com/antonino.nocera.35</tns:p_t>
  </tns:Me>
 </tns:SocialEdge>
...
</tns:SocialGraph>
```

Fig. 2.10: An example of an XML document.

Lines 6-20 show the definition of the Twitter profiles of two persons. Lines 22-37 represent the definition of a new bundle and a new resource of type “photo” (line 33). In lines 39-44, a *follow* edge among the two Twitter profiles is defined. The bundle described in lines 22-28 is published by a profile and the *publishing* action is encoded in lines 46-51. The resource is contained in the bundle as shown in lines 53-58 with the definition of a *containing* edge. Finally, the second account in Facebook of one of the authors is modeled in lines 59-66 and the information that this account and that of Twitter belong to the same person is encoded in lines 67-72.

2.5 Case studies

Evaluating the accuracy of a model is a difficult task because often a golden standard misses [35]. In these cases, evaluation can be done by humans (e.g., [175, 165]) or by applying the model to an application and evaluating the results (e.g., [196]). In this section, following the latter approach, we describe how our model has been profitably applied to two applications very relevant in the context of social network analysis. The first application regards the extraction of information from a multiple-social-network scenario, the second one concerns a particular analysis done on social network data.

2.5.1 Information extraction

It is well known that any analysis activity on social network users needs a preliminary task implementing the extraction of data from social networks. In the past, several crawler-based strategies have been adopted to extract data, such as Breadth First Search [247], Random Walk [164] or Metropolis- Hastings Random Walk [225].

A crawling task should implement the following steps.

1. Selecting the starting account (seed). This step is very important to provide data useful to the specified application. Usually, the starting account is randomly selected from an available pool of accounts. For particular analysis, the seed can be selected from those accounts having some characteristics, for example, being a power user (i.e., they have a number of contacts much higher than the average user [162]).
2. Building the sub-graph. In this step, the information about this account is created: it includes the user account, contacts, published resources, and so on. This step is strongly dependent on the data model used.
3. Selecting the next account. There exist several strategies to implement this step. A first possibility is to randomly select another profile (uniform sampling), and

Algorithm 1 *BFS*

```
Input  $p_0$: a seed profile
Variable  ProfileQueue: a FIFO queue of profiles
 1: ProfileQueue:=∅
 2: insert $p_0$ into ProfileQueue
 3: while ProfileQueue !=∅ do
 4:     poll a profile $p_s$ from the ProfileQueue
 5:     store $p_s$
 6:     store follow edges of $(p_s, p_t)$
 7:     for each follow edge $(p_s, p_t)$ do
 8:         insert $p_t$ into ProfileQueue
 9:         store $p_t$
10:     end for
11:     store me edges of $(p_s, p_t)$
12:     for each me edge $(p_s, p_t)$ do
13:         insert $p_t$ into ProfileQueue
14:         store $p_t$
15:     end for
16:     store resources $r_s$ of $p_s$
17:     store bundles $b_s$ of $p_s$
18:     store containing edges $(b_s, r_s)$
19:     store publishing edges $(p_s, b_s)$
20:     store shared edges $(b_s, b_t)$
21:     store tagging edges $(p_s, br_s)$
22:     store referencing edges $(b_s, p_t)$
23:     store like edges $(p_s, pbr_t)$
24: end while
```

this is feasible whenever a social network uses an identifier for accounts and the domain of identifiers is known and limited. This occurs for example for Facebook and Twitter [109]. Another possibility consists in selecting one profile (i.e., a node of the graph) connected with the last visited profile by a *follow* edge or a *me* edge (see, for example, [164, 225]). Again, it is also possible to select more than one (even all) among the profiles referred above, as done for example in [247]. Once one or more profiles have been selected, Steps 2 and 3 are iterated until the desired amount of data have been extracted or a stop condition has been reached.

As it emerges from the previous description of the crawling task, data extraction is not simple to be performed because there is the problem of receiving data from different sources. In this case, we need a model that is able to handle indifferently data from different social networks.

For instance, if we need a Breadth First Search crawler operating over multiple social networks, our model allows us to implement the solution described by Algorithm 1.

The algorithm performs a Breadth First Search over user profiles and gathers all the information related to each profile visited. Specifically, it starts by adding the seed profile p_0 to the FIFO queue and executes a loop until the queue has any element. At each iteration, first, it extracts a profile from the queue (Line 4), and all its information are gathered from its social network (Line 5). Then, according to the Breadth First Search logic, the visit continues by considering the neighbors of the currently visited profile. For this purpose, all current-profile neighbors (i.e., the profiles linked by means of a *follow* edge) are stored and added to the FIFO queue (Lines 6-10). Moreover, because we are operating in a multiple social network scenario, also the other profiles of the considered user in other social networks are visited. This is obtained by storing and adding all the profiles linked by means of a *me* edge to the FIFO queue (Lines 11-15). Finally, all the information related to the actions performed by the current account (i.e., profile) are also stored (Line 16-23). The store operations are implemented according to the technicalities presented in Section 2.4, which regards how information is extracted from social networks. For the sake of presentation, such technicalities are not explicitly reported in the algorithm.

Observe that the algorithm presented above can be simplified: indeed, it is possible to set the maximum number of iterations performed by the algorithm and to reduce the information gathered for a profile on the basis of that strictly requested by the application built on top of the crawler. The former goal can be obtained by modifying the loop-stop condition at Line 3 of the previous algorithm, whereas the latter goal can be fulfilled by removing the unnecessary instructions at Lines 16-23.

It is worth mentioning that a solution very close to the one described above has been implemented in the SNAKE system [64], a tool supporting the extraction of data from social network accounts. In this system, our model has been successfully used to allow the extraction and storage of the information about the neighborhood and the alternative accounts in other social networks of a seed profile.

In order to test the correctness and effectiveness of the data extraction procedure above, we designed a prototype implementing the techniques described in Section 2.4. After the prototype implementation phase, in which we fixed several errors in data extraction, we performed a final prototype validation step[1]. For this purpose, we gathered a controlled dataset from real-life social network accounts. We run a social network crawler (as described in Section 2.5.1) starting from a randomly selected seed. As for seed selection, we observe that many social networks, such as Facebook and Twitter, associate each account with an incremental integer (specifically, a 64-bit number). Thanks to this feature, to obtain a random seed it suffices to generate numbers uniformly at random in a suitable interval and, for each number,

[1] Clearly, changes in the technologies used by the social networks (e.g., API) should need suitable adaptations of the prototype.

	User Profiles	Resources	Tags	Comments	Friendships	`me` edges	Likes	Posts
Twitter	12	22	41	28	81	35	15	31
Facebook	14	26	35	23	78	29	11	18
LinkedIn	11	21	34	17	54	24	13	21
Flickr	9	28	39	19	42	18	9	23
Google+	13	29	40	31	80	31	18	28
LiveJournal	9	14	21	14	36	16	N.A.	14
Advogato	8	11	N.A.	12	N.A.	11	N.A.	12
about.me	8	N.A.	11	N.A.	N.A.	9	6	10

Table 2.1: Dataset composition: number of data per feature and social network (N.A. indicates that a feature is not available for that social network).

to verify whether it corresponds to an existing account (because an account could have been deleted). Fortunately, the most of the identifiers are associated with active accounts, so that few attempts are enough to find a valid seed. In the implementation of the crawler, to have a feasible stop condition, we slightly modified Algorithm 1 by enforcing that when the `ProfileQueue` (see variable of contains Algorithm 1) contains 200 profiles, no more profile is inserted, so that the crawler ends by visiting such profiles.

At the end of the crawling activity, we obtained a dataset containing information (concerning profiles, resources, friendship, etc.) coming from several social networks. Observe that not all profiles included in the `ProfileQueue` appear in the dataset because some of them were private or inactive. Specifically, the dataset is composed of 265 data from Twitter, 234 from Facebook, 195 from LinkedIn, 196 from Flickr, 270 from Google+, 124 from LiveJournal, 54 from Advogato, and 44 from about.me. Table 2.1 reports the number of data extracted for each feature subdivided by social network.

We analyzed this dataset to find possible extraction faults, due to many practical issues in processing the social network data. We manually verified that the information extracted in the dataset corresponded to the actual online data. In Table 2.2, we explicit the checked attributes for each type of data. We considered *correct* the operation of extraction of a profile, a resource, and so on, if and only if the value of the extracted attributes matched that actually reported on the Web. In particular:

- a profile is considered correctly extracted if the data about the url, the belonging social network, and the screen-name match the corresponding values on the Web;
- for resources, the verification test regards the url and the resource type;
- the correctness of a tag is verified by checking the source profile, the target resource or bundle, and the label itself;

Type of data	User Profiles	Resources	Tags	Comments
Checked	Screen name	url	source profile	source profile
Attributes	OSN name	resource type	target	resource uri
	url		label	

Type of data	Friendships	`me` edges	Likes	Posts
Checked	source profile	source profile	source profile	source profile
Attributes	target profile	target profile	entity	referred resource

Table 2.2: The attributes compared to verify the correctness of the extracted information.

- a comment is considered correct if the source profile and the content (i.e., the bundle uri) match the online ones;
- the correctness of friendship and `me` edges involves the target and the source profiles;
- to assess if a like is correct, we checked the source profile and the entity (i.e., a resource, a bundle, or a profile) to which the like referred to;
- and, finally, we verified the correctness of a post by analyzing the source profile and the bundle or the resource which it refers to.

The evaluation showed that the whole dataset has been extracted with no error: therefore, the number of successful tests reached 100% at the final stage of the prototype development.

Finally, we recall that the comparison among our model and the state of the art approaches on the basis of the concepts and relationships to extract has been discussed in Section 2.6 (see, in particular, Table 2.4).

2.5.2 Matching accounts on social networks

The problem of matching user accounts is receiving a great attention in several application scenarios, such as personalization [69, 128, 248]. Indeed, people have accounts on diverse social networks, where they disseminate several traits of their personality. Gathering these traits in a unique profile is extremely useful in disparate application contexts. Unfortunately, automatically connecting the different social-network identities of users is not a trivial task due to the heterogeneity of the users' information representation in different social networks.

A common approach to address this problem utilizes profile matching techniques typically based on a set of identification properties, such as username, to find correspondences between user identities. For instance, the authors of [128] describe a technique relying on usernames and tags used in three collaborative tagging systems: Flickr, Delicious and StumbleUpon. In particular, for the identification of candidate users to be matched across social systems, the authors extract information about tagging activities to measure the frequency of use of each tag. Then, they compare tagging behavior in the different social networks to identify candidates. Finally, they extract information about the usernames of these candidates and use popular string similarity functions to match them. Another approach is that of [235], which makes use of machine learning classification techniques to match user profiles on the basis of user real name, birthday, interests, attended high school, job, and so forth, extracted from social networks. Moreover, they consolidate their results by analyzing the structural properties of the direct-friend network. Another approach is described in [248], in which the authors infer 7 hypotheses on the relationships among the usernames chosen by a single person in different communities, starting from the observation of data in BlogCatalog. They propose an approach that, given a username u suitably extracted from a source community, and a target community c, generates a set of candidate usernames in c corresponding to u. The approach first generates a set of usernames from u by adding and removing suitable prefixes and suffixes. Then, it exploits the information extracted through a Web search on Google aimed at checking for the existence of each candidate username in such a way as to reduce the returned set of usernames. Also the approach of [180] can be collocated in this context, even if its purpose is little different. Indeed, in this paper the authors focus on the de-anonymization of a network by using auxiliary information from a different social site on the basis of membership overlap and structural similarity. The authors give a demonstration of their solution on Flickr and Twitter. For this purpose, first, they extract and anonymize information from Twitter. After this, they extract information from Flickr and use it to apply their de-anonymization strategy on the modified Twitter graph.

All the above approaches require the comparison of profile information coming from different social networks, which could be quite heterogeneous. Handling such an heterogeneity is a very hard task and requires a pre-phase in which data extraction and concept-matching have to be performed. In this context, the adoption of our model can be very useful because it avoids this preliminary step.

Observe that, in the context of identity matching, our model has been successfully adopted in [63]. The approach proposed computes the similarity between two accounts belonging to two different social networks by combining two contributions: a string similarity between the usernames of the two accounts and a contribution based on a suitable notion of common-neighbors similarity. The latter component leads to a recursive definition of the overall inter-social-network similarity. Indeed, the common-neighbors notion has to rely on the same notion because neighbors be-

Name	*Sensitivity*
Salton Index [207]	0.01
Jaccard Index [129]	0.01
Sorensen Index [218]	0.01
Hub Promoted Index [199]	0.00
Hub Depressed Index [166]	0.01
Leicht-Holme-Newman Index [154]	0.01
Resource Allocation Index [253]	0.01
Local Path Index [253]	0.03
Our model applied to [63]	0.87

Table 2.3: Comparison among our approach and the state of the art.

long to different social networks, and, hence, common nodes have to be detected too.

The identification of matching accounts can be seen from two different points of view leading to the formulation of two sub-problems. The former concerns the identification of accounts of the same user in different social networks starting from a seed account. This is an intrinsically-hard task due to the wideness of the search space, which could potentially involve the entire social Web. The latter, instead, receives two accounts and aims at verifying whether these belong to the same user.

The use of our model allowed us to simplify these issues and to handle all profiles in a uniform way. We carried out an analysis of the performance of the identity matching approach in [63] adopting our model: concerning the first sub-problem, starting from a seed account of a user in a social network, the approach was able to find, among all the user accounts of the other covered social networks (which is a set of more than 10^9, i.e., almost all Web users), the alternative accounts of the same user in 57% of cases in a time interval ranging from 1 to 2.1 seconds.

As for the second sub-problem, we compared the identity matching approach in [63] adopting our model with the most meaningful local and quasi-local similarity indices (namely, Salton Index, Jaccard Index, Sorensen Index, Hub Promoted Index, Hub Depressed Index, Leicht-Holme-Newman Index, Resource Allocation Index, and Local Path Index). In our evaluation, we preliminarily extracted information about a set M of 100 social accounts, each having a secondary account in another social network. Then, we run each of the above techniques and obtained a set M' of the accounts that the technique detected as matching accounts. Clearly, M' represents a set of true positives. Finally, we measured the *sensitivity* of the techniques as the ratio $\frac{|M'|}{|M|}$. Table 2.3 summarizes the results of this comparison.

The results of Table 2.3 show that, when applied to our application domain, our approach outperforms classical state-of-the-art approaches based on common neighbors techniques. Indeed, while those approaches have a sensitivity less than or equal to 0.03, our approach reaches a sensitivity of 0.87.

It is worth noting that the low performance of common-neighbors-based approaches can be justified by the fact that they detect as similar only accounts having exactly the same information. However, in the considered scenario, typically users create accounts on social networks for different purposes and, thus, they can share different contents. Due to this reason, the intersection of neighbors can be low. As discussed previously, to overcome this problem, our approach does not rely on syntactic intersection but on similarity-based intersection. For the interested reader, further and deepened details on these experiments and comparisons are available in [63].

In summary, we can state that the success of this technique strongly relies on the model described in this chapter.

2.6 Related work

Traditionally, social networks have been mainly represented through two kinds of mathematical tools: matrices and graphs. These structures allow the modeling of information about tie patterns among social actors.

The approaches that adopt matrices representation to model social networks [153, 223] belong to the second group. Specifically, the approach of [153] incorporates social influence processes in the specification of a weight matrix W, whereas the approach of [223] uses a tensor to model the interaction between resources and users.

Examples of the second group are: Kronecker graphs model [155]; the class of model networks presented in [182], which are generalizations of the much-studied random graph of Erdös and Rényi [95] to model social networks; the approach of [204], which tries to model users interests through an *interest map* obtained by partitioning an individual's social graph and others, such as [68, 239], which model their application scenarios with graphs.

The hypergraph theory [137] allows a hyperedge to connect an arbitrary number of vertices instead of two in regular graphs. For instance, Ghoshal et al. [108] introduce a random hypergraph model to describe the ternary relationship among one user, one resource and one tag, thus making the model more flexible in the representation of many peculiar properties of folksonomies.

Other approaches adopt suitable models with the purpose of creating global user profiles by means of deep analyses of their behavior accessing multiple social networks. Often, the application scenarios of these approaches are those of ontologies and folksonomies.

Still in the field of ontologies, [88, 175] present ontology-based applications concerning social aspects. In particular, [88] deals with team building, whereas the author of [175] formulates an abstract model of semantic-social networks, in the form of a tripartite graph of persons, concepts and instances. Hence, incorporating actors in this model, he extends the traditional concept of ontologies (composed by concepts and instances). Because the referring scenario of [175] is that of folksonomies, the adopted model represents only one action (i.e., *tagging*). It is defined as a ternary association between user, concept and object. More in detail, the set of shared object and the set of keywords defined by users themselves are extracted from social networks. These collections are, then, used to obtain the emergence of a community-based ontologies.

Other interesting approaches in this context are [223, 103]. For instance, [223] proposes a cross-tagging approach, whose goal is to create a system capable of improving the set of tags of a social site with the tags used in the other sites. The enriched set of tags allows two main applications: the automatic annotation of resources, which were not originally labelled and the enrichment of user profiles. As a side effect, the more refined profiles introduce an higher precision in the computation of user similarities.

Some studies focus on the problem of integrating data of different social sites [186] and [111]. In [186], the authors propose an approach that gathers data about user activities on social sites. Suitable ontologies are used both to analyze these data and model user interests. In [111] the authors provide an unsupervised method for integrating multiple data views of a user in a single social network to produce a unified graph. They carry out this task using a form of rank aggregation applied to nearest neighbor sets.

Other recent works take advantage of social network modeling like [106, 25]. In [106], the authors try to find a method to model and simulate interactive behavior in OSNs. Their aim is to predict what users post or reply with regard to sentiments and to analyze how information spreads across the network. Finally, a comparative analysis of four mining tools based on social network graphs is presented in [25]. These tools can easily model the structure of social networks.

The system proposed in [192] has some relation with our own, as a set of meta-APIs working on social networks is provided. However, while the approach of [192] aims at creating external web services allowing the retrieval of information coming from different social networks via a unique and comprehensive platform, our approach, instead, provides a framework focused on social network users (thus, with a

	User	Resource	Tag	Comments	Friendship	Multi Networks	Like	Post
[155]	√	–	–	–	√	–	–	–
[250]	√	√	√	–	√	–	–	–
[223]	√	√	√	√	–	√	–	√
[219]	√	√	√	–	–	√	–	√
Our approach	√	√	√	√	√	√	√	√

Table 2.4: A comparative analysis of our model.

user-centered vision) allowing the aggregation of the information concerning a user coming from all the social networks he belongs to. The technical counterpart of the above feature is that our approach, besides APIs used also in [192], relies on further technologies, such as FOAF, XFN and HTML parsing. Moreover, new (user-centered) entities are considered, like `me` edges, which link two accounts belonging to a single physical person. As a consequence, differently from [192], the aim of our chapter is to support the development of models and languages for user-centered social-network-based programming in large, according to software engineering principles of genericity and polymorphism.

Our approach has some common aspects with these proposals. However, none of them consider the possibility of integrating information coming from different and heterogeneous social networks. This additional feature makes our model strongly different from the approaches presented above, because the uniform representation of all the peculiarities of different social networks is a non-trivial task and needs ad-hoc solutions to be pursued. Indeed, the solution adopted by our approach is to build a suitable middleware on top of social networks to support internetworking applications. To accomplish this task, we create a complete XML model, mapping all social network actions with abstract concepts. Therefore, the approach followed in this chapter is practical, as we solve the trade-off between complexity/expressiveness of the conceptual model and implementation issues in favor of the latter. The resulting benefits from the implementation perspective appears considerable.

To show the significance of our contribution, we pose this question. To reach the goal of this chapter, could one of the models proposed in the literature be used, even though it is defined for different purposes? Obviously, if the answer is yes the significance of our proposal is compromised. Among the models proposed in the literature analyzed to answer the above question, we choose [155] and [250], which are the most recent and representative models using graphs and hypergraphs, respectively.

In addition, we consider [223] and [219] because they are the most suitable models based on matrices and ontologies, respectively.

We compare our model with those mentioned above with respect to the following functionalities:

1. modeling of user accounts and/or profile information (e.g., screen name, user picture, etc.);
2. representing Web assets such as photos, external links, videos, etc.;
3. supporting the labeling of social entities (user profiles, resources, and so on) with tags;
4. modeling the comments expressed by users on social entities;
5. storing information about user relationships (e.g., friendship);
6. distinguishing information coming from different social networks (multiple social networks);
7. supporting the *I Like it*, by means of which users express that they like, enjoy or support a given content;
8. storing information about *who posts what*.

3

Privacy in Social Networks: a crucial issue

Understanding user behavior in Online Social Networks (OSNs) is an important challenge in the field of social network analysis, especially when it comes to privacy. So far, many studies considering only one OSN or, at most, comparing results obtained for a single OSN, have been provided. Nowadays, users typically join more OSNs and this is an important aspect that should be taken into account. In this chapter, we give a relevant contribution in this direction, by analyzing the behavior of users having accounts on both Facebook *and* Twitter*. This way, the analysis is well-founded because it is conducted on a common set of users and, further, a number of specific investigations become possible (as studies on common friendship). Our research is carried out on data extracted from the Web, and allows us to find important specificities of these kind of users concerning their privacy setting, the choice of friends and their activity in general. Then we focus on the robustness of social network privacy settings and describe an interesting case study inherent to a possible attack to Facebook privacy mechanism.*

3.1 Materials and Methods

We aim to compare people's behavior in the two most popular social networks, which are Facebook and Twitter. We base our analysis on the concept of membership overlap, to study a number of behavioral aspects.

The first one is about privacy and disclosure of personal information. Recent studies on Facebook have shown that both a strong association between low engagement and privacy concern and a significant relationship between privacy awareness and privacy concerns/self-disclosure [255] exist. Our study aims to answer the question "Is there a connection between user awareness about privacy threats and membership overlap between Twitter and Facebook?".

The second aspect we study is about friendship. OSNs are important for maintaining social relations and previous studies have found that friendship is positively correlated with bridging social capital. As for this aspect, we study what is the attitude of users to have friendship relations overlapping between Facebook and Twitter and if a correlation between number of friends in Twitter and Facebook exists.

The last issue we deal with concerns the activity of users belonging to both Twitter and Facebook. [193] found that the prime goal of user activity on Facebook is to self-promote or to maintain relationships, whereas other studies showed that some types of activity are a sign of narcissism [203]. Our study aims to answer the question "What about user activity and how the prevalence of activity on Facebook or Twitter is correlated to membership overlap?".

Observe that, the specificity of our analysis is to consider accounts of the same person in Twitter and Facebook in order to draw conclusions on the use of the two social networks by highlighting similarities and differences. In the following, we provide the detail to allow full reproducibility of all experiments.

Data Extraction. Information necessary for our analysis has been extracted from Twitter and Facebook from January to May 2014, adopting the model presented in Chapter 2.

An important issue in the extraction of our data is the need to detect whether two accounts belong to the same person. Fortunately, users can explicitly declare connections from the profile of a social network to another by means of special links, called me edges. From a technological points of view, there exist several ways to extract this information from the account of a user in a social network. The most common leverages on XFN (*XHTML Friends Network*) which is an HTML microformat allowing for the representation of the kind of relationship existing between two user accounts. This is obtained by empowering the set of values that the rel attribute of the HTML tag <a> (which represents a link) can assume. In our case, we focus on the value "me" (rel=`me') which indicates that the corresponding link represents a me edge. Another common way for extracting information on me edges relies, once again, on the use of social network APIs already mentioned above.

Data Sampling. The major problem in this task was the need of collecting data about user accounts that have a me edge from a social network to the other. To do this, visiting a social graph by any existing crawling technique results in biased data, thus it is not suitable. Indeed, it has been deeply studied that classical techniques for sampling a social graph, such as Breadth First Search and Random Walk, produce samples biased in the node degree distribution and newest sampling strategies, such as Metropolis-Hasting random walk and re-weighted random walk, solve the above problem of node degree distribution but still produce samples with very few *me* edges, as proved in [62].

As a consequence, we decided to perform uniform sampling, as it has been referred as the ground truth technique for obtaining unbiased social network datasets

[110]. Uniform sampling is not a trivial task in general. However, for Facebook and Twitter, this activity is facilitated by how user identifiers are defined. Indeed, both adopt 64-bit identifiers for user accounts. In particular, the URL address of the profile page of a Facebook (resp. Twitter) user is http://www.facebook.com/YYY (resp., http://twitter.com/account/redirect_by_id?id=YYY), where YYY is a 64-bit positive integer. Thus, to obtain a uniform sample, it suffices to generate numbers uniformly at random in a suitable interval and, for each number, to verify whether it corresponds to an existing account (because an account could have been deleted).

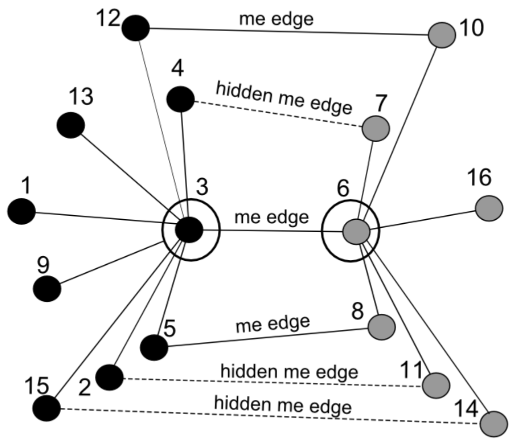
Extraction, Transformation and Loading of data. As we were interested in data about users who have accounts in both social networks, uniform sampling has been executed as follows. We started by uniformly sampling Twitter to collect a set of 875 Twitter users declaring (by a me edge) an account also in Facebook. Then, we proceeded by visiting their Facebook accounts and we found that 118 of them were not valid URLs, therefore we cleaned up our dataset by removing these nodes. For the 757 remaining Twitter bridges, we gathered information about their alternative accounts on Facebook and about their direct neighbors. The dataset obtained is composed by the following tables: user, friend and me. The first table contains the fields: screen-name, indegree, outdegree, sn_id, social network, visited, public. The first attribute screen-name is the account name chosen by the user at the moment of the registration; indegree represents the number of *followers* of the user; outdegree, instead, is the number of *followings* of the user[1]; sn_id is the original social network identification for the user; the field social network can assume two values, namely Facebook or Twitter, specifying the referring social network; finally, the attributes visited, public are two binary values indicating whether the profile has been directly visited or if it has been found as friend of a visited profile, and whether the profile has accessible information (public) or not. The table friend represents the social graph, i.e. maps the friendships of the accounts sampled, whereas table me contains information about me edges. After the collection of rough data from the previous steps, we need to preprocess data by removing duplicates and accounts with not valid URLs and by generating tables indexes and keys. To support our analysis, in a second round, we added two additional fields at the table users, namely tweet count and creation date. These new attributes have not null values only for Twitter users.

Observe that, in a previous study, [61] showed that, due to disparate reasons, users often do not declare *me* edges explicitly and proposed an algorithm to infer

[1] Due to the symmetric nature of the friendship relationship in Facebook, indegree and outdegree have the same value for Facebook users.

hidden me edges between two accounts. Hence, we built a further table, namely hidden me with the results of the application of the technique described by [61], to find further pairs of accounts associated with the same user.

Dataset Description. In order to better understand the aspects described above (e.g., me edges, hidden me edges, etc.), a portion of our dataset related to a user is sketched in Figure 3.1: black and gray nodes are user accounts of Twitter and Facebook, respectively, and the real name of the account is also reported[2]. Specifically, we consider a user having an account on Twitter (node 3) who declared a me edge to his Facebook account (node 6) and his social network friends (neighborhoods). In this case, also some neighbor accounts are overlapping: in particular, nodes 12 and 10, nodes 5 and 8, belong to the same user because it is explicitly declared by means of me edges; whereas, nodes 4 and 7, nodes 2 and 11, nodes 15 and 14, have been found to be of the same user by the algorithm proposed by [61], and, therefore, are labeled as hidden me edges (i.e., not explicitly declared).



Id	Screen-name
1	tatahsantana
2	neltonpyter
3	Pedao_Vini
4	denioneto
5	Raafinha_mlk
6	pedro.vinicios.372
7	denio.neto
8	Raafinha.S
9	TwitPic
10	carlos.junior.50767
11	neltonpyter2
12	Carlos_Juniiior
13	instagrao
14	gabrielabritto1
15	Gabrielabritto1
16	monalisa.carla

Fig. 3.1: A fragment of our dataset.

The whole dataset is available at the address http://www.infolab.unirc.it/cihb2014.html (for the Reviewers: the password to open the archive is 081733849) and some statistics are reported in Table 3.1.

[2] Data were not anonymized at this stage to preserve full reproducibility of experiments. In case of publication, we will do this.

Table 3.1: Some statistics of our dataset.

Seen Nodes	$\sim 4 \cdot 10^6$
Visited Nodes	304,715
Twitter *Nodes*	*158,755*
Facebook *Nodes*	*145,960*
Visited Edges	368,314
Bridges	910
Twitter Id Range	[1 - 359999]
Indegree Range	[0 - 52,295,363]
Outdegree Range	[0 - 2,436,264]
Bridge Indegree Range	[1 - 238,523]
Outdegree Range	[1 - 99,843]

3.2 Results

In this section, we perform a number of experiments on the collected sample to answer the questions presented in the introduction.

3.2.1 Privacy setting

The first analysis concerns the choice users about the privacy level in Facebook. We investigate if users who have two accounts (in Twitter and Facebook) show the same behavior as other users when it comes of privacy concerns. Therefore, in this experiment the control variable is the user having two accounts, whereas the dependent variable is the privacy setting. We count how many users of the sample with two accounts choose to disclose their Facebook information on the social network, thus making their Facebook account public[3]. We obtain that about 87% users kept their Facebook account private, and this result is statistically valid with a 95% confidence level and 2.25% margin of error.

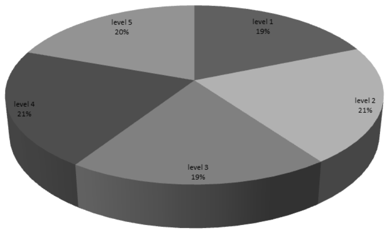
Moreover, we analyze if there are some differences, in terms of privacy setting, among users with different number of friends (i.e., degree). In this case, we consider also the degree as independent variable. To perform this analysis, we discretize degree by applying the logarithmic binning function reported in Table 3.2. The choice of the logarithmic binning function allows us to obtain almost equal-width bins due to the well-known power law distribution of node degree.

In Figure 3.2, we report the distribution of the users with private account according to their discretized degree (indegree and outdegree in Figure 3.2.(a) and

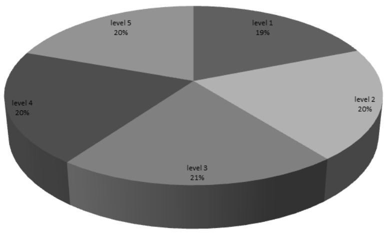
[3] This analysis is limited to Facebook because Twitter accounts cannot be private.

Table 3.2: The logarithmic binning function used to discretize degree.

Value	Bin
$x < 10$	1
$10 \leq x < 100$	2
$100 \leq x < 1000$	3
$1000 \leq x < 10,000$	4
$x \geq 10,000$	5



(a) *Indegree*



(b) *Outdegree*

Fig. 3.2: Private account distribution on the basis of their degree.

(b), respectively). We observe that there are no significant differences among the five degree intervals considered in our experiment: indeed, about one fifth of private accounts are in each of the five bins. We can conclude that the number of friends (i.e., the node degree) does not seem to affect the choice of having a private account.

3.2.2 Friend overlap

In this section, we study the attitude of users to have overlapping friendship relations. In particular, we analyze if users with account both in `Twitter` and `Facebook` have overlapping neighborhoods in these two social networks (i.e., how often they add the same person as friend both in `Twitter` and `Facebook`).

In this experiment, given a user u with account in both `Twitter` and `Facebook`, we define the metric CFF_T (*Common Twitter Friend Fraction*), which measures the fraction of the friends of u in `Twitter` who are also friends of u in `Facebook`. Analogously, we define CFF_F[4], which considers `Facebook` instead of `Twitter`. To detect if a friend is the same in the two social networks, we check for the presence of a `me` edge between these two accounts, as described in Section 3.1. Observe that it could occur that a friend has not explicitly declared the `me` edge and this could lead to

[4] Clearly, $CFF_T \neq CFF_F$ because the initial user set from which the fraction is computed is different.

Table 3.3: Computing the friend overlap in Twitter and Facebook.

	M	SD
CFF_T	0.028	0.048
CFF_F	0.007	0.015
CFF_T^*	0.075	0.074
CFF_F^*	0.029	0.049

underestimate the friend overlap. To overcome this problem, we use the approach proposed by [61] for discovering not declared me edges, which allows us to detect (with a good approximation) also these overlapping friends. We denote by CFF^* the results of the computation of friend overlapping obtained by extending the set of common accounts with the approach proposed by [61]. Therefore, in this experiment we consider the user with two accounts and the social network as control variables; whereas *Common Friend Fraction* is the dependent variable.

The result of this analysis is reported in Table 3.3, in which the metrics defined above were summed and averaged and the standard deviation is also computed. This experiment shows that there is no significant overlap among the friends of the two accounts of a user in Twitter and Facebook. Indeed, the overlap is only about 7% in Twitter on average, and the overlap measured on Twitter is higher than on Facebook.

3.2.3 Friend distribution

The aim of this section is to study the relation between number of friends and membership overlap between Twitter and Facebook, by means of three experiments.

In the first experiment, we consider each user with account both in Facebook and Twitter and compare the number of friends he has in Twitter and Facebook. Therefore, the control variables of this study are the user with two accounts and the social network, whereas the dependent variables are the indegree and outdegree of users. The result of this measure is reported in Table 3.4, which shows the average value of indegree and outdegree of Twitter accounts versus the degree of Facebook accounts of the same users. Observe that, for Facebook, indegree and outdegree coincide because of the symmetric friendship relation. This table shows that the average degree of Twitter accounts is much higher than that of Facebook. However, because it is well-know that degree in social networks follows a power law distribution [37], we need to better investigate this results.

For this purpose, in the second experiment, we compute the median value (i.e., the central value separating the higher half of degree values from the lower half)

Table 3.4: Average number of Twitter and Facebook friends.

	Twitter	Facebook
Indegree	1626.92	133.71
Outdegree	587.43	133.71

Table 3.5: The median of indegree and outdegree for the four sets of users.

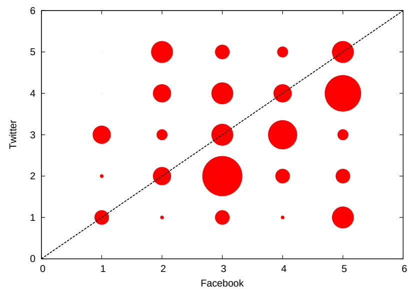
	indegree	outdegree
declaredT	61	114
otherT	53	108
declaredF	726	726
otherF	679	679

instead of the average degree, as the former is a more meaningful indicator of the trend of degree in case of power law distribution. We partition the users of our sample into the following four sets, and for each of them we compute the median value of degree:

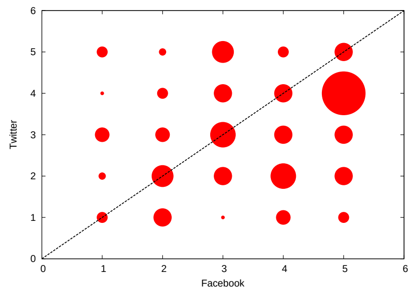
1. declaredT, composed of the Twitter users who declared to have an account also in Facebook;
2. otherT, the remaining Twitter users;
3. declaredF, composed of the Facebook users who declared to have an account also in Twitter;
4. otherF, the remaining Facebook users;

The results of this experiment are shown in Table 3.5. Combining these results with those of the previous experiment, we find that, while most of the users of Twitter have a degree lower than the users of Facebook (see Table 3.5), Twitter *power users* (i.e., the users with a very large number of friends) have a degree much higher than *power users* of Facebook (Table 3.4).

In the last experiment, we consider again a user with account both in Twitter and Facebook, and we study a possible relation between the number of friends he has in each social network. For this purpose, we discretize the indegree and outdegree in 5 equal-width bins by applying the logarithmic binning function reported in Table 3.2 and we build two dispersion matrices having the indegree (resp., outdegree) level of Twitter as Y-axis and the degree of Facebook as a X-axis. This way, we can observe if a relation between degrees of the same user in the two social networks exists. Figure 3.3 shows the graphical representation of the two dispersion matrices:



(a) *Indegree* Twitter *vs Degree* Facebook.



(b) *Outdegree* Twitter *vs Degree* Facebook.

Fig. 3.3: The scatter plot showing the relation between number of friends in Twitter and Facebook.

the more the points are next to the bisecting line, the higher the relation between the degrees. Moreover, the size of the points represents the number of occurrences of that degree combination.

From the analysis of this figure, we observe that the bigger points are in the area under the bisector, meaning that there is a slight anti-correlation between Facebook and Twitter degrees (i.e., the higher the degree of a user in Facebook, the lower the corresponding degree in Twitter).

3.2.4 User Activity

Through the experiments described in this section, we study the relation about membership overlap and user activity. Specifically, the purpose of this analysis is to investigate the behavior of three different typologies of users: (1) declared, which are users who have an account both in Facebook and Twitter and explicitly declared this, (2) hidden, which are users who have an account both in Facebook and Twitter but did not declare this[5], (3) other, which are the remaining users.

For this analysis, we define the normalized activity coefficient NAC of a Twitter account as t_c/y_a, where t_c is the number of tweet posted and y_a is the number of years since the account has been created. As done for degree in Section 3.2.1, we discretize NAC by applying the logarithmic binning function reported in Table 3.2 and we compute its value for each typology of users. In this study, the user typology is our control variable, whereas the activity coefficient is the dependent variable. The obtained results are reported in Table 3.6, in which the first row represents the mode (i.e., the bin which most of the users fall in) for the specific type of user. From these results, we find that users who have more accounts are less active than the others

[5] As done in Section 3.2.1, such users are detected by using the approach defined by [61].

Table 3.6: The frequency distribution of the normalized activity coefficient for the different types of user.

(a) Declared users.

NAC bin	%
3	32.4
1	27.5
2	24.9
4	13.8
5	1.3

(b) Hidden users.

NAC bin	%
1	91.3
3	3.8
4	3.6
2	0.7
5	0.3

(c) Other users.

NAC bin	%
4	46.1
3	34.0
5	9.0
2	7.2
1	3.4

who have account only in Twitter. By looking at Figure 3.4, which shows a different view of the results, we conclude that, among users with more accounts, those who have a not declared Facebook account are very inactive.

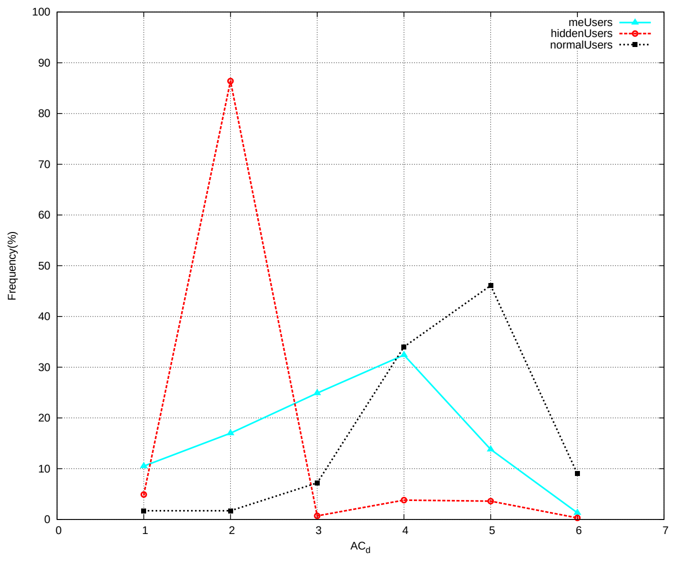


Fig. 3.4: The frequency distribution of the normalized activity coefficient for the different types of user.

3.3 Discussion

The goal of our work is to analyze the behavior of users belonging to more social networks. The chosen setting refers to the two most popular social networks, which are

Facebook and Twitter. In this section, we highlight and discuss our major findings and issues.

The first analysis we have carried out concerns privacy awareness. We have found that 87% users with account in Facebook and Twitter chose to keep their Facebook information private (i.e., they have a private accounts and their information are accessible only to their friends). Moreover, we observed that this percentage is not significantly affected by the number of friends they have. As precedent studies have shown that only 52% Facebook users have private accounts [87], we conclude that the users declaring to have an account both on Facebook and Twitter pay more attention to privacy issues. These users are able to declare a secondary account (i.e., by a me edge), know technological aspects of the social network, and are aware of privacy setting. As a consequence, they may be more accurate in the definition of the privacy policy for their accounts. The first important result of our study is that privacy awareness has a positive impact on privacy value.

The second result we found is that there is no significant overlap among the friends of a user in Twitter and Facebook. This behavior can be explained by considering that, in general, users create multiple accounts in social networks for different purposes (e.g., sport profiles, music profile, job profiles, etc.). Thus, each account is associated with a specific interest or part of the user life and the friends they add comprise people related to the specific context which the profile refers to. A recent study by [122] showed that personality differences between Facebook and Twitter users exist, such that more sociable individuals gravitate towards Facebook, while less sociable ones gravitate towards Twitter, thus making friendship overlap less likely. Our result confirms the study done by [190] according to which Facebook users tend to add as friend people they know in real life in order to transform latent to weak ties [93], whereas Twitter use is driven primarily by interest for entertainment news, celebrity news, and sports news [115]. Moreover, our findings confirm also the hypothesis that the notion of online friend can comprise different kinds of friendship.

The third analysis concerned the number of friends of Twitter and Facebook users. The result obtained about the median and average number of friends in Facebook is consistent with that reported in the study of [32] and confirms that the number of (stable) friends is bounded (around 150) due to limitations of the human brain [90]. We found that, while most of the users of Twitter have a degree lower than the users of Facebook, Twitter *power users* have a degree much higher than *power users* of Facebook (Table 3.4). This can be explained according to the theory that follow relationships in Twitter are typically towards important and famous people who act as *power users* [115], so that there are few users but with a very large degree. In contrast, in Facebook, friends are often personally known [93] so that their number is limited. Finally, our result about anti-correlation between degree in Twitter and Facebook allows us to conclude that a user joining both Twitter and Facebook does not equally subdivide his activity between the two social networks, but has a preference for one. From the analysis of our sample, we observed that such users prefer Facebook as main platform.

The findings of the last experiment is that users who have more accounts are less active than the others who have account only in Twitter. This can be explained by considering that the latter users may focus their attention only in one social network, thus directing all their posting activities on it. Vice versa, the users posting contents in at least two social networks (i.e., Twitter and Facebook), concentrate the total amount of posts in one of them. Moreover, we found that users who do not declare to have more accounts are the least active and appear "lazy". The importance of this result is related to several aspects of user behavior: it has been found that people who are active on social networks are more likely to feel connected [232], that they are high in ICT innovativeness [252] and that social activity may enhance social presence and increase social influence [74].

3.4 Limitations

Social network analysis has assumed an extraordinary importance since the birth of online social networks, because they target and record social relationships in the most complete and detailed way among all digital services. As a consequence, they are a huge source of information about people, whose analysis may result in strategic and useful knowledge. The multiple-social-network perspective is a promising approach compliant with the evolution of social web. We have addressed the problem of comparing the behavior of a user in Twitter and Facebook, believing that the multiplicity of social networks is an important aspect to take into strict consideration when studying the complex phenomenon of OSNs. The novelty of our work is to have approached the comparison from a truly multiple-social-network perspective, according to which users with profiles in both OSNs have been considered as object of study, to have meaningful and well-founded results. This study has some limitations. First, it considers only Facebook and Twitter among the numerous online social networks, so that our results can not be generalized to other or all social networks. However, we chose the two most popular ones, thus ensuring the validity of results for a very large part of social network users. The next limitation concerns the extraction of the sample from the Web, because only public information can be retrieved, and the limited size of the sample. Another limitation is that, like many other observational studies, we cannot draw causal conclusions. To overcome these limitations, future work could extend the sample size both considering other OSNs and using other extraction techniques, such as surveys, yet taking into account that this method results in a strong sampling bias and makes it difficult to acquire large samples. Moreover, because some results could be explained by other latent variables, such as age, location, sex, these should be incorporate in the future analysis.

3.5 A threat to privacy in Facebook: a case study

As stated before, the rapid growth of social networks, primarily Facebook, has coincided with an increasing concern over personal privacy. This explains why more and more users personalize their Facebook privacy settings. As a matter of fact, the list of friends is often one of the profile sections kept private, meaning that this information is perceived as sensible. In this section, we study the robustness of this Facebook privacy protection feature, showing that it can be broken even in the less advantageous conditions for the adversary. To do this, we exploit both the potential information extracted from user alter accounts in Twitter and a social network property, recently demonstrated for Twitter, called interest assortativity.

The preliminary experimental results reported in this section, give a first evidence of the effectiveness of our attack, which succeeds even in the most difficult case that is when the information about the victim are minimum.

3.5.1 Approach formalization

In this section, we describe the technique we propose to discover (at least a part of) friends of a social network user who decided to make private his friend list. First, we observe that our approach works for social networks, such as Facebook, in which friendship relations are symmetric (that is, if the user u_1 is friend of u_2, then also u_2 is friend of u_1).

The intuition underlying our approach is that privacy setting of an account is indicated by the account owner, meaning that u_1 can choose to make private his friendship with u_2, whereas u_2 can choose to make public his friendship with u_1. Consequently, by looking at u_2's account, the friendship between u_1 and u_2 can be inferred even thought u_1 tries to hide it. It is worth noting that the mere execution of the strategy sketched above has a strong limitation that makes this trivial search unfeasible. Indeed, due to the huge number of social network accounts, the search space of possible friends is limitless. Moreover, this strategy returns at most one friend for each possible friend analyzed.

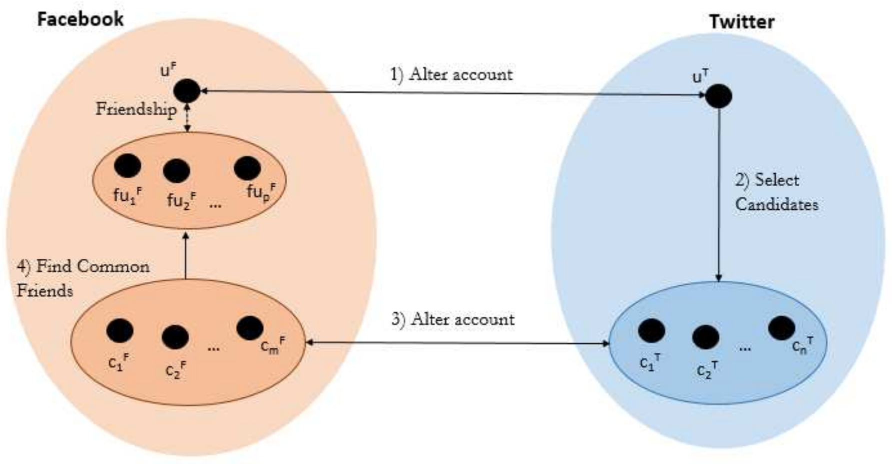


Fig. 3.5: Graphical representation of the approach.

To overcome these drawbacks, we designed a technique more sophisticated than the above one to reach two important advantages. The first one is to provide a relatively limited number of accounts to analyze (say *candidates*), thus reducing the search space and making this solution feasible. The second advantage is that, thanks to a suitable selection of each candidate, the processing of each candidate account is able to return more friends of the initial account u_1 (to obtain this, we exploit the mechanism of friend community present in social networks).

The technique used to discover private friends basically relies on three procedures, *find alter accounts*, *select candidates*, and *find common friends*. For the sake of presentation, we describe at high level how our proposal works, whereas the detailed implementations of the above three procedures are provided in Sections 3.5.1, 3.5.1, and 3.5.1, respectively.

The input of our technique is u^F, which is the account of the user u in a social network F that supports symmetric friendship relations. In the follow, we instantiate F with Facebook, the most popular social network. Clearly, the friend list of u^F is private. The output of our technique is a set of accounts $f_i u^F$ that are friends of u^F in Facebook. Our approach is schematized in Figure 3.5 and consists of 4 steps.

1. In order to discover private friends of u^F, the first step we run is *finding alter accounts* of u^F. This step aims to identify a secondary account of u in another social network (how to perform this task is described in Section 3.5.1).
   It is well know that users register account on different social networks and use them for different purposes. Among all the social networks in which u has registered an account, we are interested in his secondary account in Twitter: indeed, Twitter is a very famous and common social networks, which is used to exchange

very short messages. At the end of this step, we obtain u^T, which is the account of the user u in Twitter.

2. Now, we run the second step of our technique, that is *select candidates*. This steps aims to identify Twitter accounts that are in the same community of u and that can lead to discover the private friends of u^F. Among the three steps, this is the core one and is deeply explained in Section 3.5.1. Let $c_1^T, \ldots, c_n^T$ be the set of candidates outputted at this step.
3. In this step, we run the procedure *finding alter accounts* for each candidate c_j^T, in order to discover his/her account on Facebook, say c_j^F. At the end of this step, we have found some accounts on Facebook that hopefully are in the same Facebook community of u^F (because they are alter accounts of users in the same community of u in Twitter).
4. In the last step, for each c_j^F, we run the procedure *finding common friends* in order to find the list of friends in common between c_j^F and u^F. This procedure, which can appear *magic*, is instead provided by many social networks in order to give members the opportunity to find new friends. The detail on the implementation of this step is given in Section 3.5.1 At the end of this step, we obtain the set $f_i u^F$ containing some (hopefully all) private friends of u^F.

From the high-level description of how our technique works, it is clear we cannot guarantee that this approach is always able to break the privateness of friend list: indeed, it is necessary that alter accounts are found and friend community exists. However, as we will show in Section 3.5.2, we experimented that for many real-life accounts, the execution of this technique is able to discover at least a portion of private friend list.

In the next sections, we will describe the implementation of the three procedures used in our technique.

Finding alter accounts

Many social networks provide their users with the possibility to add in their own profiles a link toward one of their accounts in another social site or external website. This feature is typically enabled during the creation of the user profile. This information is extremely useful in our approach because it allows the identification of the accounts belonging to the same person in a multi-social network scenario.

Technically speaking, users who explicitly declare their alter accounts via social network tools (also known as `me` edges), physically create special links among social networks. The basic strategy leverages the use of social network APIs, a set of methods and services, typically available for social network developers, allowing the interaction with social-network data and functionalities to create new software on top of them.

However, not all social networks provides APIs to extract this information. Therefore, another possibility to extract alter accounts relies on the XFN (*XHTML Friends Network*) standard, an HTML microformat to represent relationship among user accounts. This is obtained by empowering the set of values that the `rel` attribute of the HTML tag <a> (which represents a link) can assume. In particular, the value "me" (i.e., `rel=`me'`) is used to indicate that the corresponding <a> link represents a `me` edge.

Finally, in cases in which users do not declare explicitly their alter accounts, several approaches proposed by the scientific community can be applied to detect missing `me` edges.

For instance, the approach of [61] makes use of account similarity to detect a missing `me` edge in case a suitable threshold is exceeded. The similarity between two accounts is obtained by combining two contributions: one computed as string similarity between the account usernames, and one based on a suitable recursive notion of common-neighbor similarity. By leveraging this technique, we are able to extend our solution also to the cases in which users do not declare their `me` edge explicitly.

In summary, alter accounts can be retrieved by using social network APIs, XFN, or techniques such as those defined in [61, 142]. Observe that, in the case of Facebook and Twitter, XFN is adopted for declaring alter accounts.

Selecting Candidates

Once we have collected a set of nodes holding both a Facebook and a Twitter profile, for each of this users, say u, we focus on his Twitter friend list. As explained in Section 3.5.1, the next step that our algorithm performs is the selection of some suitable nodes, whose neighbors can be potentially friends of u in Facebook.

The approach we follow to obtain the set of candidates leverages the concept of *assortativity* in social networks. Assortativity is an empirical measure describing a positive correlation in personal attributes of people socially connected with each other [183]. Hence, if a network is assortative with respect to a given attribute, it means that the majority of its users tend to act as their friends when it comes of the aspect expressed by that particular attribute.

It is proved that Twitter shows assortative behavior with respect to user interests [60], where interest assortativity is defined as the preference for friends to share the same interest (e.g., sport, music). Indeed, in Twitter there exist accounts belonging to *public figures*, which, due to their influence w.r.t. a specific topic, act as a sort of representative for that topic. This way, the abstract concept of interest (or topic) can be mapped to the concrete entity of a public figure.

Thus assortativity is quantified as the difference between the measure of the studied trait in the observed network and that computed in the corresponding *random* graph of the network [183]. Therefore, we need to measure these two quantities. Let us start from the observed network.

Definition 3.1. *Given a social network G, an interest I, a positive integer t, and a node $n \in N_{t,I}$, we define the* Interest Friend Fraction *towards n of I as:*

$$IFF_{n,I} = \frac{\left|\{a \in \Gamma_{in}(n) \mid \exists\ b \in \Gamma_{in}(n)\ \wedge\ b \in \Gamma_{out}(n)\}\right|}{\left|\Gamma_{in}(n)\right|}$$

In words, given a node n belonging to the category of the interest I, IFF measures the fraction of the nodes followers of n having at least one friend b that is follower of n too.

Definition 3.2. Given a social network G, an interest I and a positive integer t, we define the *General Interest Friend Fraction* towards the most t nodes of I as

$$GIFF_{t,I} = \frac{\left|\{a \in N \mid \exists\ b \in \Gamma_{out}(n),\ n_1, n_2 \in N_{t,I}\ \wedge\ a \in \overleftarrow{\Gamma(n_1)}\ \wedge\ b \in \overleftarrow{\Gamma(n_2)}\}\right|}{\left|\cup_{\,n \in N_{t,I}} \Gamma_{in}(n)\right|}$$

In this case, we have n nodes in the category of the interest, and GIFF measures the fraction of the nodes interested in any of the t nodes that have at least one edge towards another node b that is interested in any of the t nodes too.

After measuring the phenomenon in the observed network by IFF, we need to characterize the *random* graph of the network, commonly denoted as *null model* [183]. This graph models the case in which no assortativity occurs.

Definition 3.3. *The* null model *of G is the random graph $\hat{G} = \langle N, \hat{E} \rangle$ such that for each $v \in N$, it holds that:*

1) $|\{(x,w) \in E \mid x = v\}| = |\{(x,w) \in \hat{E} \mid x = v\}|$;

and

2) $|\{(w,x) \in E \mid x = v\}| = |\{(w,x) \in \hat{E} \mid x = v\}|$;

In words, it is obtained by maintaining the nodes of G and by replacing the deterministic occurrence of edges by a random variable in such a way that in-degree and out-degree of nodes is maintained.

Now, we define how to measure IFF in the null model.

Definition 3.4. *Given an interest I, a node $n \in N_{t,I}$ for any t, IFF in $\hat{G}$ is computed as:*

$$\widehat{IFF_{n,I}} = \mathcal{P}\Big(y \in \Gamma_{in}(n) \mid x \in \Gamma_{in}(n) \wedge y \in \Gamma_{out}(x)\Big)$$

where $\mathcal{P}(X)$ stands for probability of X.

The following theorem allows us to compute this probability.

Theorem 3.5. Given an interest I, and a node $n \in N_{t,I}$ for any t, then:

$$\widehat{IFF_{n,I}} = \sum_{d=1}^{u} \frac{|N_d|}{|N|} \cdot \Big(1 - \mathcal{P}\big(|N| - 2, d-1, |\Gamma_{in}(n)| - 1\big)\Big) \tag{3.1}$$

where:

$\mathcal{P}(c,a,b) = \prod_{k=0}^{b} \frac{c-a-k}{c-k}$

and N is the number of nodes, N_d is the number of nodes with degree d, and $\Gamma_{in}(n)$ is the set of followers of n.

Proof (sketch). $\widehat{IFF_{n,I}}$ is obtained as the sum for any degree $1 \le d \le u$ of the product of two factors: the probability of having a node with degree d, and the probability that this node and at least one of its friend follow the same public figure. The second factor is computed knowing that the probability that a node with degree a does not have as friend one of b nodes selected in N (i.e., $\mathcal{P}(|N|, a, b)$) follows a hypergeometric distribution and, thus:

$$\mathcal{P}(c,a,b) = \frac{\begin{pmatrix} b \\ 0 \end{pmatrix} \cdot \begin{pmatrix} c-b \\ a \end{pmatrix}}{\begin{pmatrix} c \\ a \end{pmatrix}} = \prod_{k=0..b} \frac{c-a-k}{c-k}$$

□

Now we are ready to give the formal definition of Interest Assortativity.

Definition 3.6. *Given an interest I and a node $n \in N_{t,I}$ for any t, we define the* Interest Assortativity *towards n of I as:*

$$IA_{n,I} = IFF_{n,I} - \widehat{IFF_{n,I}}$$

This measure gives us an index of how much a social network is biased w.r.t. the null model in terms of probability of finding friends sharing the same interest. The higher the value of assortativity, the higher the correlation in being friends and sharing the same interest.

Thanks to the assortative behavior of Twitter users, we can find people belonging to his clique by searching on the neighbors of a public figure of interest for a given user, obtaining a set of suitable candidates for our technique.

Finding common friends

By following the reasoning described in Section 3.5.1, we can extract a set of Twitter accounts, which are potentially "close" to the user, say u, for whom we want to reconstruct the friend list. Now, for each of the candidates obtained we can adopt the strategy described in Section 3.5.1 to find their alter-accounts in Facebook, thus obtaining a set of Facebook-candidate accounts.

At this point, if these candidates have a public friend list, we can verify whether u is present in any of these lists. Each time we find an account in any of these lists, due to the bidirectional nature of Facebook friendships, we have discovered an element of the friend list of u. Because the set of candidates may be very big, it is very likely that we can reconstruct the entire friend list of u. However, this strategy would require a very large number of checks to verify the membership of u in any of the friend list of the candidates.

Fortunately, Facebook provides a very powerfull set of APIs, namely Graph API [14], that can give us some advantages in our objective. Graph APIs are a low-level HTTP-based APIs useful to retrieve data from Facebook.

Specifically, it is possible to discover the mutual friends between two Facebook users by performing an HTTPS request at the following link:

```
https://www.facebook.com/friendship/<screen-name1>/<screen-name2>
```

where the parameters `<screen-name1>` and `<screen-name2>` represent the string ids used to univocally identify the two users in Facebook. Interestingly, this API method works also if one of the two accounts (that of u in our case) has the friend list private. Due to the tendency of Facebook users to form cliques [233] the above API method allows us to discover a very large set of items of the friend list of u with a single call. This reduces the number of operations required to reconstruct the entire friend list.

3.5.2 Preliminary evaluation

Our experiments were performed on a machine equipped with a 2 Quad-Core E5440 processor and 16 GB of RAM. The operating system was Linux Ubuntu Server 14.04.4 LTS, with kernel version 4.2.0-35, Java Virtual Machine version 1.8.0 45 (64-Bit) and Twitter4J as external library for Twitter API support. We wrote our implementations in Java.

To obtain the initial set of Facebook profiles to test our approach performance, we could not rely on existing datasets as they do not provide information about `me` edges. To extract necessary data, we exploited the SNAKE system [64] which allows the extraction of profile contact information from a very large set of social networks. One of the main issues in this activity was the detection of the profiles showing the right features for our investigation (i.e., an alter account in Twitter and a private friend list). However, using one of the classical crawling technique is not suitable for two main reasons:

1. First, the percentage of accounts with a `me` edge is very low [62], so it is extremely difficult to find these particular users. This implies that an almost complete visit should be performed to obtain necessary information. However, full structural information of the network is not needed, because we are interested only in Facebook users with an alter account in Twitter.
2. Secondly, a crawling technique may privilege the visit of some nodes with particular structural properties (i.e., very high degree) introducing some biases in our results.

As a consequence, we decided to perform uniform sampling. Although, uniform sampling is not a trivial task in general, for Facebook and Twitter, it is facilitated by how user identifiers are organized. Indeed, both social networks adopt 64-bit identifiers for user accounts. However, because we are looking for private Facebook accounts, we cannot start our sampling from Facebook. We started by uniformly sampling Twitter to collect accounts having a `me` edge towards Facebook[6]. In particular, the URL address of the profile page of Twitter has the following structure: `http://twitter.com/account/redirect_by_id?id=xxx` , where `xxx` is a 64-bit positive integer. Hence, to obtain a uniform sampling, we generated numbers uniformly at random in a suitable interval and then we checked if it corresponds to a real account with a `me` edge towards Facebook and whose alter account in Facebook has a private friend list. From this sampling step we obtained a set of 355 accounts.

We proceeded by analyzing the set of Twitter friends for each of the accounts above, and we selected public figures among them. In our case, because the set of candidates (see Section 3.5.1) is a subset of followers of these public figures, we neglected those public figures who have a too high in-degree and considered only accounts with an in-degree ranging from 500 to 1500. Clearly, this choice was made only to guarantee computation feasibility, and did not affect the performance of our technique as will be shown in the following.

After this, we considered the Facebook alter accounts of the candidates, extracted in the previous step, and continued by calling the Graph API method described in Section 3.5.1 to

[6] Observe that, the direction of the `me` edge is not relevant for our objective, because we only need to know that the two accounts belong to the same person.

verify whether we were able to reconstruct the private friend list of the original set of 355 users. As a result, we succeeded in 259 cases, thus obtaining a probability of success of 0.73.

3.6 Related Work

With the increase in both the number and the dimension of OSNs, the development of approaches aiming to deeply investigate their main features has become welcome. A very interesting trend of the recent literature is to try to characterize user behavior in different platforms.

There are a number of relevant reasons that call for a deeply study on how users act when logging to these sites. First, analysis of user behavior allows the evaluation of the performance of existing social systems, more refined site design [65, 243] and user tailored advertisement placement policies [242]. Second, accurate models of user behavior in OSNs are crucial in applications of social studies such as viral marketing. Indeed, one of the main goals of marketers is the spread of their promotions quickly and widely. For this reason, they need to understand how users interact and to build models representing these interactions in such a way as to foresee how information will flow [156, 241]. Third, the study of user behavior helps the prediction on how much the future workload of OSNs will influence the whole Internet traffic, which is an essential information to properly dimension the Internet infrastructure and content distribution systems [145, 202].

As for the analysis about user social behavior in a social network, [228] and [130] explored search behavior on Twitter: while [228] make a deep analysis of large-scale query logs and supplemental qualitative data, [130] focus on the study of the topological and geographical properties.

A systematic measurement study on the statistics of the social network underlying the video sharing service YouTube is reported by [73], in which a deep analysis on user behavior through a number of ratings, views and comments on YouTube videos is carried out. Another study on YouTube is presented by [167] and focuses on the identification of different classes of user behavior to improve recommendation systems for advertisements in OSNs.

Several studies have looked at the comparison of the behavior of sample of users among different OSNs [113, 251, 92]. However, all these studies extract trends on the use of social sites and compare them through statistical parameters derived from the analysis of large sets of users. Thus, they do not observe the behavior of the same user in the two systems. [113] try to characterize user activities and usage patterns in some popular OSNs like Bebo, MySpace, Netlog, and Tagged. In [251], instead, the authors consider the differences between Twitter and traditional news media content. A comparison of Facebook and MySpace on the aspects of trust and privacy is reported by [92]. The obtained results show that members of both sites have similar levels of privacy concern. However, Facebook members report higher trust in both the social network itself and the other Facebook users, and are more confident to share identifying information. [214] collect objective, privacy-preserved behavior data from user that are active in both Facebook and Gmail. The authors make a comparative analysis on user behavior in OSNs and their way of using email services. The analysis shows that a large portion of social interactions still occur through email messages, whereas participants tend to be more emotional on Facebook.

Using a general population sample of 300 users, [122] examine the personality correlates (i.e., Neuroticism, Extraversion, Openness-to-Experience, Agreeableness, Conscientiousness, Sociability and Need-for-Cognition) of social and informational use of the two OSNs Facebook and Twitter. By examining also age and gender they show that personality is related to online socializing and on how people seek and/or exchange information. Moreover, a preference for Facebook or Twitter is associated with differences in personality. [99] analyze the role of persuasion in the actions that users perform in two social networking sites. The samples analyzed comprise U.S. users from Facebook and Japan users from Mixi. The authors compare the two OSNs on four persuasion goals: creating profile pages, inviting friends, responding to content by friends, and how frequently they connect to the site. Their analysis reveals the differences and similarities in how Facebook and Mixi are designed to influence users towards the achievement of these four goals.

[102] compare user behavior on Sina Weibo and Twitter, by analyzing how people access microblogs and by comparing the writing style of Sina Weibo and Twitter through textual features of microposts. Moreover, based on semantics, they study and compare the topics and sentiment polarities of posts of the two systems. Finally, they investigate the temporal dynamics of the microblogging behavior such as the drift of user interests over time. [132] study user behavior from four different perspectives, *connection* and *interaction*, *traffic activity*, *mobile social behavior*, and *malicious behaviors* in order to classify attacks (such as spam and Sybil attack) on the basis of the severity level of security threat.

Our approach differs from those presented in this section because, while all of them base their analysis on samples derived from different social networks and generalize on user behavior by observing statistical aggregative parameters, our analysis, instead, is carried out on a set of pairs of accounts (of Twitter and Facebook, respectively) such that the accounts of each pair belong to the same user. Hence, our study actually reflects the different way of being and acting of the same person in the two considered social networks.

4

A security problem: Social Network Mobile Apps

More and more often social networks are used as information and collaboration platforms worldwide. This trend is also underlined by a great number of applications requiring access to data from such systems to enrich the functionalities provided. With the increasing of this tendency, the problem of access control becomes a crucial issue. Above all, the lack of fine-grained rules when applications use social network APIs to access user profiles is a problem of great interest. For instance, in Twitter, the supported access control policy is basically on/off, so that if a (third party) application needs the right to write just a message on behalf of a user, he is enforced to grant access with no restriction to his whole profile. This enables a large set of security threats and can make (even inexpert) users reluctant to run these applications. To overcome this problem, we propose an effective dynamic solution working for Android Twitter applications based on a middleware approach. The proposed technique enables other possible benefits, such as, anomaly-based malware detection through API-call patterns. Moreover, due to its modularity, this solution can be directly extended to a multi-social network scenario.

4.1 Background

In this section, we provide the technical knowledge necessary to understand our proposal. It concerns the authentication mechanisms on social networks.

Most of online social networks have embraced the paradigm of Software as Service (SaaS) meaning that all features are available to end users as a service through the Internet. The services offered to users are delivered as a set of Application Programming Interfaces (APIs) available via the http/s protocol. However, one of the main issue is the necessity of providing authentication mechanisms that do not necessary require the direct log into the social network for accessing its data and functions through external services. For this purpose, the authentication of many social networks, such as Facebook, Twitter, Google+, is based on the OAuth 1.0a protocol described in RFC 5849, which has been replaced by the new OAuth 2.0 authorization framework.

Basically, OAuth allows a third-party application to gain limited access to an http/s service without users expose their password. To do this, it provides a token-based mechanism so that an application can access data on in a social network profile if a valid token is provided. Such a token is automatically delivered to the application directly from the social network after the user manually approves it. In the latest version of the Twitter API (v1.1), two forms of authentication, both based on Oauth 1.0a, are allowed: *Application–Only* Authentication, where only the application is authenticated and acts without using a user context, and *Application–Users* Authentication, which allows the application to act on the user's behalf. The former is the basic authentication and allows it to call APIs and retrieve public information on the social network. Clearly, in this case, the application cannot obtain any private information of any user and cannot perform actions, such as tweeting, on the user's behalf on Twitter. Now, we focus our attention and technical discussion on the case of Twitter. From a technical point of view, Twitter authentication scheme is based on the use of two Twitter API keys, namely the *consumer key* and the *consumer secret*. Specifically, the consumer key is used to identify the application itself, whereas the consumer secret, in combination with the consumer key, allows the application to perform authenticated requests on its behalf.

The detailed flow is as follows: First, the application encodes its consumer key and secret and performs an https POST request to the link `https://api.twitter.com/oauth2/token` to send these credentials to the `oauth2 / token` endpoint. Then, the `oauth2 / token` validates them and replies with a *bearer token*. Finally, the application uses the bearer token for the future interactions with the Twitter APIs. Clearly, this authentication scheme does not support services or data requiring a user context. The Application–Users Authentication is the most common authentication scheme and works by authenticating both the application (or external service) and the user. The mechanism allows users to explicitly declare their willing to provide the application with a token for opening a context on their behalf. This step is allowed only if the application is registered on Twitter and, hence, is equipped with a valid consumer key. The basic requirement for this authentication scheme is an *access token*, which allows the application to operate on behalf of the user whom this access token belongs to. There are mainly three authentication mechanisms to obtain the user token and are related to the specific functionality the application wants to provide. Specifically, the possible authentication mechanisms are: *Sign in with Twitter*, *3-legged OAuth* and *PIN-based OAuth*. As for the first mechanism, in this case the application uses Twitter account information to identify a user, thus allowing a fast log into its services. The application performs a POST requests, using its consumer information as credentials, to the oauth/request_token endpoint via the link `https://api.twitter.com/oauth/request_token`. This request must embed as parameter a callback URL where the user is redirected after the authentication on Twitter. In the response, a *request token* is returned, which is used as a GET parameter to redirect the user (via browser) to the link `https://api.twitter.com/oauth/authenticate`. This way, the user can authenticate on Twitter, which, if the login procedure succeeds, sends a response containing a *verify token* univocally associated with the request token to the callback URL. Finally, to obtain a valid *access token*, allowing the retrieval of the account information for the user via the `account/verify_credentials` endpoint, the application performs a POST request using the verify token as parameter to link `https://api.twitter.com/oauth/access_token`. The `account/verify_credentials` endpoint can be accessed through the link `https://api.twitter.com/1.1/account/verify_credentials.json`.

The second authentication mechanism *3-legged OAuth* is used when an application wants the privileges to act on the user's behalf. To do this, the user is redirected on Twitter to authorize the application. This mechanism is almost identical to the previous one. The only significant difference is that the access token obtained with this procedure can only be used once. In the case the application cannot directly make use of a browser to obtain the access token, it can use the *PIN-based OAuth*. Also this mechanism is similar to the first. However, since the application cannot access a Web browser, a URL is returned by Twitter in the last step of the procedure. This is shown to the user who can now use a browser to access this link and authenticate on Twitter. If the authentication succeeds, than a Web page with a 7-digit code is shown to the user. Now, the user has to input this code to the application (that should be waiting for this input) and the application can complete the procedure to obtain the access token by sending a final request to the Twitter endpoint `oauth/access_token` with the 7-digit code as parameter.

Once the application has been authenticated, Twitter can grant it one of the following privileges (chosen before the token generation): *(1) Read only*, meaning that the application can only read information on the user timeline and account *(2) Read & Write*, meaning that the application can perform all reading activities along with writing/deleting contents and profile statuses as well as sending direct messages to other users. Clearly, privilege 2 subsumes privilege 1.

4.2 Problem formulation and desiderata

In this section, we motivate the problem by real-life examples and sketch the characteristics that a solution of this problem should have. To better illustrate how our approach works, we make explicit reference to a scenario in which a mobile application requires the access to Twitter information via services provided by the social network (according to the SaaS paradigm). This scenario is very common, as there are a lot of third-party mobile applications that offer enhanced functionalities extending Twitter basic services.

4.2.1 Motivations

The first example we present concerns the case of an application allowing users to write tweets with more than 140 characters. There are several ways to achieve this goal: by automatically creating a short URL linking the whole user tweet, or by posting the long message as an image, or by splitting the original tweet into a series of smaller linked tweets. In this case, an application offering this feature should act on behalf of the user and, therefore, should have the *Read & Write* privilege.

A more complex and interesting example of the SaaS paradigm application is given by the use of the social signature [59]. Social signature is a recent proposal implementing electronic signature by means of Twitter. This social network is used as "device" enabling the generation of the signature and also as a trusted-third-party allowing signature sharing. Signers use their mobile phone to generate the signature and to post a suitable tweet on their own account. The tweet shows the digest of the signed document along with other information necessary to guarantee authenticity and non-repudiation. In this case, the data to be posted on Twitter cannot be manually generated by the user. Again, the mobile application implementing the social signature protocol should be authorized by the user to run Twitter services with `Read & Write` privilege. The most relevant side-effect of this authorization is that, once an application obtains the consumer's access token, it may perform almost everything on the user profile, although it needs only a subset of those permissions. The examples described above clearly show situations in which the low granularity of Twitter access control model gives rise to possible security threats because the least privilege principle is not accomplished.

4.2.2 Solution Requirements

To address the problem described above, a possible solution should satisfy the following requirements:

Fine-grained privileges on Twitter services. The solution should allow the user to explicitly define the specific functionalities allowed to a mobile application on his Twitter account. Specifically, the user should be able to directly specify the set of API methods that the application is allowed to use. In the first example of Section 4.2, the user may allow the application to only use the `POST statuses/update` method instead of granting any call to other not required writing methods, such as `POST statuses/destroy/:id`, which allows the deletion of a previous tweet or `POST account/update_profile` allowing the update of personal profile information such as the user contact name.

Device-Independent policy definition. Once the user defines a content access policy to the Twitter functionalities for a given service, this should be applied to any application on each specific device that the user wants to bound with these policies. This aspect has a heavy impact on the solution architecture because access control rules must be available outside the user device.

Anomaly-based detection of malicious usage of the service. Although the user is able to specify fine-grained privileges, the misuse access privileges by an application is still possible. A valid solution should be able to detect this illegal behavior by analyzing how an application makes use of Twitter API's methods.

We will see in the following how to build a solution able to ensure such requirements.

4.3 System Architecture

In this section, we describe our solution and the general architecture of its implementation. Specifically, to satisfy all requirements described in Section 4.2.2, we identify two main components for our system: *(i)* An Android service that monitors all traffic toward Twitter from the user device; *(ii)* a remote middleware that communicates with the Android service on the user device and verifies whether the activities performed by third-party applications match the user access control rules.

A schema of the proposed framework is shown in Figure 4.1.

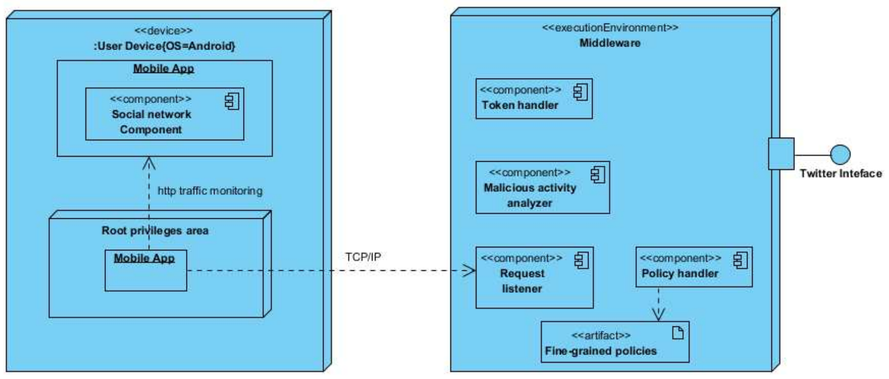


Fig. 4.1: Deployment diagram of the proposed framework.

The basic idea is that every call to a method of the Twitter APIs that is generated by a third-party application running on the user device is intercepted and analyzed. To do so, we implement an application requiring root privileges on Android and monitoring all https traffic towards Twitter. When a communication towards Twitter is identified, the flow is redirected to our remote middleware, which will act as an intermediate by performing the requests to Twitter on behalf of the application that originated the communication. The middleware implements all the functionalities to satisfy the requirements of Section 4.2.2. In the next sections, we will describe the two components in more detail.

4.3.1 The Android service

To achieve the goal our system must take control over the communication towards Twitter originated by the user device. For this purpose, we provide the device with an application, implemented as an Android service, that has to monitor all the Web traffic. However, as stated in the Android specifications [6], the execution of an Android application follows the *principle of least privilege* for which each application runs in a separated Linux process inside an independent virtual machine. Moreover, each Linux process, associated with an Android application, is executed by a different Linux user, who can access only the information in its isolated address space. The mechanism underlying this specification ensures that no direct interaction may occur between different applications. Figure 4.2 schematizes the concept outlined above.

To bypass this *sandbox* security strategy of Android, we have to give to our application the root privileges. This way, it can access all the information available and makes use of the underlying Linux operative system functionalities. The target of our root-level application is very similar to that of some other famous applications, such as [10, 18]. Observe that, the practice of rooting an Android phone is widely accepted especially for security solutions, such as [16, 17]. Moreover, it is also common to run applications that require special privileges to bypass carrier installed software, overclock and underclock the processor, modify system files, or even more trivial actions like system setting customization. However, the main difference is that our application has not only to intercept any communication towards Twitter starting from the device, but also has to redirect the communication towards our middleware server.

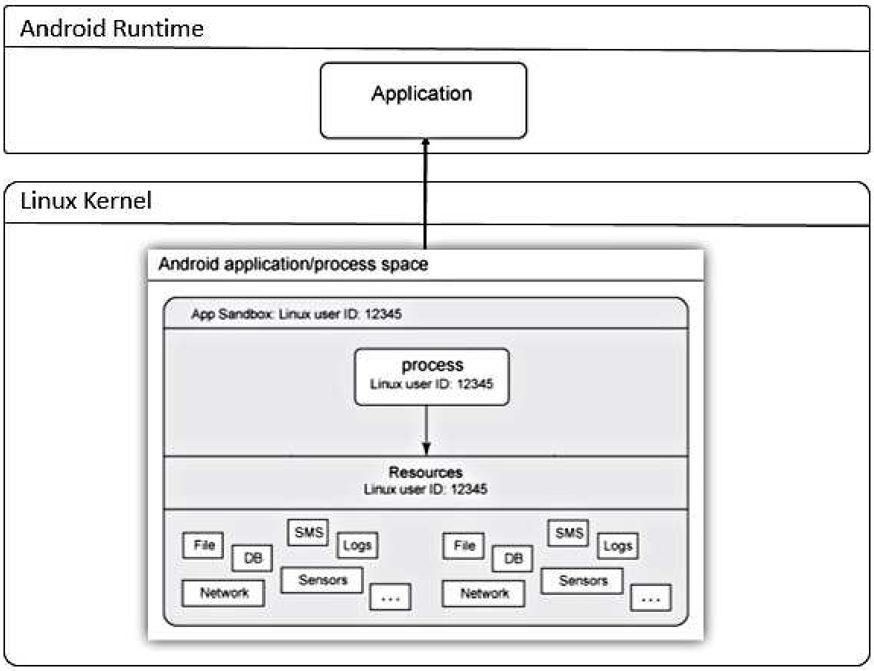


Fig. 4.2: Android application execution representation.

Our middleware will perform all the https requests to Twitter on behalf of the user application and will return responses back to the application on behalf of Twitter. To achieve this goal our application does the following: *(i)* jeopardizes the PKI trust chain by installing a self-signed fake root certificate (this is possible with root privilege from Android 4.0 on [20]); *(ii)* makes use of the powerful Linux firewall underlying the Android system, namely `iptables`. By properly setting up rules on `iptables`, it is possible to redirect any *TCP* traffic accordingly to our strategy. Observe that, even though it could be in principle considered a break of security, the jeopardization of the PKI trust chain is, in fact, a practical solution. Indeed, it is commonly adopted in supervised contexts (such as, enterprise environments), in which a proxy servers perform traffic flow analysis and monitoring [15]. To do so, we started from the source code of *DroidWall - Android Firewall* that is an open source project for a front-end application to the Linux firewall [12]. The aim of DroidWall is to restrict the set of applications that are allowed to access the device data networks, therefore we suitable modify its code to implement our solution.

4.3.2 The middleware service

This component of our system is in charge of managing the authorization policies, defined by the user, to the Twitter API's contents. Specifically, all the traffic generated by any third-party applications on any user-device is redirected, by our Android service, to this external component. In this component we have the following modules:

Request Listener. This module receives all the https requests originated by our Android service running on any device.

Token Handler. The aim of this module is to handle the interaction with the OAuth authentication protocol. In the initialization step, our application requires the access to Twitter services by issuing an *Application–Users Authentication* via OAuth protocol (see Section 8.1 for further details). The output of this procedure is that the user is prompted to authorize the application to access his Twitter account; in this phase, the application requires also the authorization type (which is either *Read only* or *Read & Write*). At the end of this procedure, in the case of positive authentication, the application is provided with a (long-term) token that can be used to speed up the authentication procedure in the next communications with Twitter. The basic strategy of our approach is to replace the original request of Application–Users Authentication performed by the third-party application on the user device with a corresponding request generated by the middleware itself. This way, the middleware will obtain the access token, hereafter referred as *master* token, from Twitter and will perform all the next requests on behalf of the third-party application. On the other side, this component generates a *sub*-token, associated with the original *master* token, that will be sent back to the application. As specified by the OAuth protocol, the application will use this *sub*-token in the call to any Twitter API's method.
Policy Handler. This module has two main functions: *(i)* to store a set of access control rules for each application and *(ii)* to provide the right mapping between any sub-token and the corresponding rules. The implementation of fine-grained policies is orthogonal to the approach described in this chapter, therefore it is possible to follow any existing approach, such as [98], even though a specific model would be desirable (this is matter of our future work). A possible example of an access policy could be the following. Consider the case in which a user wants to prevent a third application from reading his private direct messages. In this case, the Twitter APIs handling direct messaging are the following:

- `GET direct_messages/sent`. The call to this method returns the 20 latest direct messages sent by the authenticating user together with information about the sender and the recipient user.
- `GET direct_messages/show`. This method returns the direct message whose identifier is specified in the *id* parameter, along with information about the objects, the sender and the recipient.
- `POST direct_messages/destroy`. Through this API method, the application can destroy a direct message received by the authenticating user, by specifying the message identifier in the required *id* parameter.
- `POST direct_messages/new`. This API method allows the application to send a new direct message to a specific user on behalf of the authenticating user.

Clearly, the user could not want to manually specify all the API methods that has to be blocked to achieve his goal. In this case, he can give only the general command $\langle direct\_message, deny \rangle$. This module will translate this command into an XML-serialized rule, which will propagate the deny command to all the sub-elements (i.e., the API methods referring to the direct messaging feature) of the element `DirectMessage`. Clearly, because the user

```
[...]
<Policy>
<Read allow="False">
  <DirectMessage allow="False">
    <sent allow="False">
      <API>GET direct\_messages\/sent</API>
    </sent>
    <show allow="False">
    <API>GET direct\_messages\/show</API>
    </show>
  </DirectMessages>
</Read>
<Write allow="False">
  <DirectMessages allow="False">
    <Destroy allow="False">
      <API>POST direct\_messages\/destroy</API>
    </Destroy>
    <PostNew allow="False">
      <API>POST direct\_messages\/new</API>
    </PostNew>
  </DirectMessages>
</Write>
</Policy>
[...]
```

Listing 4.1: An example of a policy denying direct messaging.

did not specify whether the rule has to affect reading or writing feature, it will be applied on both of them (see Listing 4.1).

Anomaly Detector. This module implements a further security mechanism allowing the detection of malicious behavior of third-party applications. By following an anomaly-based approach, the above task is accomplished on known applications for which a behavior fingerprint is stored in the middleware. To detect anomalies this module can take advantage of the approaches proposed in [172].

Twitter Interface. It is in charge of handling the communication between the middleware and Twitter. In particular, it forges the new messages to Twitter according to user policies and receives the query results from Twitter.

4.3.3 Protocol

In this section we provide some further details about the protocol adopted by the entities involved in our system to fulfill our solution tasks.

As stated in Section 4.1, OAuth protocol works by performing an initial strong authentication which allows the association of the user and the application that will performs queries on his behalf. The output of this procedure is that the application will be provided with an access *master* token through which it can interact with protected Twitter-user resources on his behalf. Our protocol interposes, between the application that wants to use external APIs and Twitter, both an Android service and a middleware. Hence, standing in the way of all user application requests, our middleware is able to analyze and send them to Twitter only if they match user access policies. Figure 4.3 describes our protocol among Twitter, the middleware and the Android device.

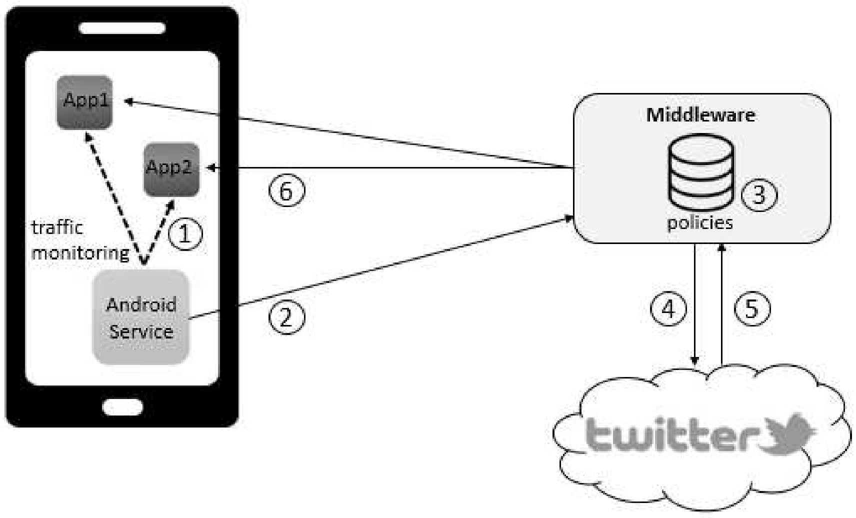


Fig. 4.3: A communication diagram of the entities involved in our system.

In the first phase, the Android service monitors the original request of Twitter APIs performed by the third-party application on the user device (1). Observe that, all the calls to methods of the twitter APIs are performed via the https protocol. Moreover, they share the same structure, i.e.: `api.twitter.com/[version]/[APImethod]/[parameters]`. Therefore, once our Android service (see 4.3) monitors all https traffic and redirect to the middleware components all traffic originally intended for Twitter(2). The listener module of the middleware receives all requests and activates the `Token Handler` and the `Policy Handler` which process this request and forward it to the `Twitter Interface` module if it matches the user policies (3). Also in this step, the `Anomaly Detector` module is activated and analyzes the request by combining it with information on previous activities from the same third-party application to identify possible malicious behaviors. After that, in the case of positive matching, this module builds a similar request and sends it to Twitter in order to access the needed user resources (4). In a second phase, data flow back by following the same path described above. Observe that, in this way, the `Policy Handler` module could perform further filters on the obtained data to match user preferences in the case the used Twitter API method would not allow the right grain level (5) (for instance, read only the tweets of a given profile). We remark that this feature could not be supported by following a simple firewall-based approach. Finally, the filtered data are sent back to the Android device (6).

4.4 Related work

Nowadays the lack of security in mobile devices is a critical issue because of the pervasive diffusion of these systems [149, 70].

To solve this problem, several researchers investigated on fine-grained policy enforcement [80, 188, 210, 67]. In [80], the authors define a system called CRePE able to enforce fine-grained policies depending on the context of the smartphone. According to the combination of the value of some variables (e.g. location, time, temperature, noise, the presence of other devices) that can define a particular context, the mobile device changes its settings and, for instance, it runs only a restricted set of applications or it switches off/on some interfaces. In [188], the authors create a framework to provide install-time permission-granting policies and runtime inter-application communication policies. This allows a specific application to change its runtime behavior, specifically: which external interface to use, which applications can access its interfaces, and how they can use its interfaces. Always in the field of fine-grained authorization, an interesting work is presented in [210]. In this paper, the authors introduce DAuth an authorization protocol that allows fine-grained control of access permissions in the context of application delegation. In particular, if an application is composed of more than one external component (i.e., different weblets), then a potential risk can occur. The countermeasure proposed in this work, is the definition of a new SDK to allow the design of applications where the external components have different policies w.r.t. the on-device ones. The authors of [67] try to unify distributed component models under a common meta-model for the purpose of creating a platform independent model for the access control patterns. Hence, the access control to application resources by the different components is handled through an orthogonal security aware middleware, thus allowing a plug and play component environment. All the approaches above assume that the applications that want to leverage their frameworks must be complaint to their protocols. This requirement is not needed by our solution that can be applied to any existing third-party application.

The problem of defining fine-grain authorization to access resources is a critical issue in all the environments in which remote sharing of application services is needed. For instance, [138] deals with the issue of guaranteeing fine-grain authorization for resource management in virtual organizations within grid environments.

As for mobile environment, the ability to install third-party applications poses serious security concerns that raise the attention of lots of researchers. In particular, the authors of [181] present a policy enforcement framework for Android called Apex. This framework allows users to selectively make decisions about permissions on their device rather than automating the decisions based on the policies of remote owners and allows finer-granular control over usage. In [94], the Kirin security service for Android is proposed. This system certifies an application according with a set of security rules, thus performing lightweight certification of applications and mitigating malware at install time. A relevant tool to perform automated security certification checking of Android applications as they are installed is SCANDROID [101]. It analyses whether data flows through an application are consistent with specifications extracted from the corresponding manifests.

Finally, since our model modifies the regular OAuth protocol flow in order to guarantee more granular choices on what an application can do with APIs it is useful to notice that, in this context, some extensions to the OAuth 2.0 authorization enabling an enhanced use of permissions are presented in [213, 77]. Specifically, the focus of [213] is to provide a mechanism that computes permission ratings based on a recommendation model leveraging previous user decisions and application requests in order to enhance user privacy.

5

Trust and Online Communities: the case of TripAdvisor

Many real-life reputation models suffer from classical drawbacks making the systems where they are used vulnerable to users' misbehavior. TripAdvisor is a good example of this problem. Indeed, despite its popularity, the weakness of its reputation model is resulting in loss of credibility and growth of legal disputes. This chapter aims at describing a reputation model abstractly considering service providers, users and feedbacks, and implementing the theoretical notion of certified reputation to concretely define a strategy to normalize *feedback scores towards reliable values. We apply this model to the case of TripAdvisor, by proposing a solution to improve its dependability neither increasing its invasiveness nor reducing its usability. Moreover, it fully guarantees backward compatibility.*

5.1 Background

In this section, we recall the main attacks on reputation systems, on which our proposal operates.

Self-Promoting attacks, concerning the possibility that a service provider increases its reputation by fake positive feedbacks. Typically, such attacks are contrasted by excluding positive feedbacks but this solution is not applicable in all real-life cases because the elimination of positive feedbacks would compromise the effectiveness of the reputation system.

Slandering attacks, which aim at falsely decreasing the reputation of other service providers (typically competitors) by providing fake negative feedbacks.

These attacks usually exploit real users who are disposed to send false feedbacks (positive or negative) about a service provider, just for friendship, or sometimes for money or competitive advantage. Often, the feedback does not correspond to a real transaction with the service provider. Unfortunately, parallel markets of feedbacks exist, in which (more or less legal) companies collect users and offer their (fake) feedbacks to any bidder. Furthermore, fake accounts can be created to achieve the same goal, if the system do not prevent this possibility. As a matter of fact, the account registration procedure of real-life systems does not implement specific features able to trust digital identities.

In these cases, *Sybil attacks* may be done to make a fake account seemly real. A sybil attack generates interaction between false accounts to mutually increase their reputation in the system.

Finally, *Whitewashing* attacks exploit the difficulty to establish users identities: in this case, a user, who wasted his reputation, creates a new account to change its reputation to the default initial trustworthiness value.

5.2 The Reputation Model

In this section, we describe our reputation model. The component of the model are:

1. A set of users U,
2. a set of service providers S, and,
3. for each service provider $s \in S$, a list of feedbacks $R(s)$, each corresponding to a transaction. A feedback $r_s \in R(S)$ for the service provider s is a tuple $\langle u, d, k, t, I \rangle$, where u is the author the feedback, d is the time of the feedback, k is the score given by u on s, t is a text motivating k and I is the set of additional resources.

The transaction corresponding to a feedback r_s is denoted by t_r. To implement the notion of certified reputation in this model, for a given feedback r_s, we need two basic measures, which are both numbers ranging in the interval $[0,1]$: (1) the trustworthiness of the identity of u, denoted by $trust(u)$, and (2) the trustworthiness of the transaction t_r, denoted by $trust(t_r)$.

The first measure $trust(u)$ is related to identity management issues and is a measure of the level of assurance of identity proofing given by the registration phase into the reputation system according to the model proposed by NIST [66]. Observe that we do not consider the level of assurance of the authentication phase, thus assuming that no attacks on accounts of users occur. This measure may take into account also external information associated with the digital identity of the user in the system (for an instance, information coming from different sources as online social networks). We remark that the trustworthiness of the identity of u is directly related to some of the threats described in Section 5.1 (for example, Sybil attacks), but it is in general related to misbehaving users, as any malicious activity is simplified when the level of assurance of user identity is low.

Equally important is the trustworthiness of the transaction. Again, many of the attacks described in Section 5.1 are based on the possibility to send a feedback about a service never used. The trustworthiness of the transaction $trust(t_r)$ is a function $h(c, w, p)$, where c is the content of the feedback (so, it is a function of text t and resources I), w measures the presence of witnesses (as theoretically provided in [124]) relying also to the time d, and p measures the trustworthiness of the presence of a confirmation about the occurrence of the transaction.

On the basis of the two measures above we are able to reach our original goal, which is the measure of trustworthiness $trust(r_s)$ of a feedback $r_s = \langle u, d, k, t, I \rangle$ associated with a transaction t_r, as the function $trust(r_s) = f(trust(u), trust(t_r))$ such that the higher $trust(u)$ and $trust(t_r)$, the higher $trust(r)$. Once $trust(r_s)$ has been computed, the score k of the feedback r_s can be corrected on the basis of the overall trustworthiness $trust(r_s)$, using a function g, tending to bring the score k to the average score obtained by the service provider s the more the value $trust(r_s)$ is low. We obtain thus the *normalized* feedback r_s^* simply as $r_s^* = \langle u, d, g(k), t, I \rangle$.

Observe that the trust model we consider in this chapter does not include any reputation mechanism regarding authors of feedbacks, but only service providers. Indeed, our intention is to better isolate the problem we intend to deal with. Obviously, further reputation mechanisms can be combined.

5.3 Application to TripAdvisor

In this section we apply the model presented in Section 5.2 to the case of TripAdvisor. Indeed, the attacks to reputation systems described in Section 5.1 fully apply to the case of Tripdvisor, as its reputation system is really weak. Fake reviews can be easily included into the system as user's identity is weakly managed, and no proof exists that a real interaction between reviewers and operators occurred. Moreover, it would be easy for an operator claiming to be victim of slandering attacks coming from competitor operators. This open the possibility of disputes difficult to solve because TripAdvisor does not provide compensation mechanisms: a user cannot provide proofs of his interaction with the operator and an operator cannot be claim to be victim of an attack by a competitor. Interestingly, as recently reported by *The Guardian* [229] the Italian antitrust authority has fined TripAdvisor €500,000 (US $600,000) following complaints of improper business practices lodged by a national hoteliers association and a consumer protection agency. The antitrust authority said in a statement that TripAdvisor had failed to adopt controls to prevent false reviews, while at the same time promoting the site content as authentic and genuine. Thus, the chosen application domain is both suitable to our model and very topical.

As a first step we have to map all the entities of the model to the TriAdvisor setting.

The set of users U is the set of travellers, potentially covering all web users. The set of service providers S corresponds to the set of restaurants, bars, hotels, and other operators registered to TripAdvisor. A list of reviews is associated with each operator $s \in S$. It corresponds to the list of feedbacks $R(s)$ of our model. Indeed, a review r_s is authored by a user u and is provided by: a date d, an aggregate score k (also detailed into different evaluation dimensions), a text t describing the interaction t_r with the operator aimed at motivating the score k, and, optionally, a set I of images portraying the experience. Thus, a review can be mapped to a feedback $\langle u,d,k,t,I\rangle$.

As presented in Section 5.2, our goal is to compute the value

$$trust(r_s) = f(trust(u), trust(t_r)) \tag{5.1}$$

for a given review r_s given by the user u, in order to compute the *normalized* review $r_s^* = \langle u,d,g(k),t,i\rangle$, once the functions allowing to compute $trust(u)$ and $trust(t_r)$ are defined.

Let us start with the computation of $trust(u)$, representing a level of assurance of identity proofing of the customer u who have authored the review. Of course, a significative part of the current weakness of the reputation system of TripAdvisor is based on the weakness of its digital identity management system.

Concerning $trust(u)$, we adopt an approach similar to the classification given by NIST about the identity proofing [66] classifying 4 levels of assurance (LOAs): (1) claimed identity with no proof, (2) presentation of credentials with no verification, (3) presentation of credentials with verification, and (4) in-person presentation only. As for the case of TripAdvisor level 4 is of course unrealistic, we consider 3 levels assigning the values $trust(u) = 0, 0.5, 1$ for them, respectively. Currently, the registration phase of TripAdvisor does not force the user to provide any non-self declared credential. Anyway, the possibility of registering via Facebook/Twitter/Google+ profile is also allowed. We propose to include in the system also a *token-based* two-factor registration mechanism, by sending a token as credential via sms or instant messaging systems uniquely associated with a phone number (such as, for example, Whatsapp or Viber). The usage of a phone-based mechanism as second factor of the initial registration increases the level of assurance w.r.t. the other options. Thus, we set $trust(u) = 0$ if the registration of u was self-declared-credential based, $trust(u) = 0.5$ if the registration of u was social-network-profile based; and $trust(u) = 1$ if the registration of u was token-by-phone based. In principle, $trust(u) = 1$ also for social-network-profile based registrations provided that the social-network registration mechanism is token-by-phone based too (as it happens for some online social networks). Observe that an indication that a given account is fake could be given also by the number of reviews of that user. Reasonably, an account created just to implement a self promoting (or a slandering) review, will have one or a few reviews. Conversely, a user showing a high number of reviews is less likely author of fake reviews. However, this aspect lies outside our focus, as it concerns a reputation mechanism regarding authors of feedbacks and the estimation of their expertise. As mentioned earlier, similar strategies (actually already naively included in TripAdvisor), could be also fortified and combined with those aspects that are subject of our work.

Concerning $trust(t_r)$, i.e., the trustworthiness of the transaction, we first recall that $trust(t_r) = h(c, w, p)$, where c is the content of the feedback (so, it is a function of text t and resources I), w measures the presence of witnesses (possibly relying also to the time d), and p measures the trustworthiness of the presence of a confirmation about the occurrence of the transaction.

Currently, no mechanism is explicitly given by TripAdvisor to *certify* a transaction. In our model we identify three pieces of information that can be used to this purpose: content, witnesses, confirmation. Let us see now if these pieces of information can be drawn from the TripAdvisor system.

Concerning c (i.e., content), consider that the review is equipped with a motivating text and, optionally, with images. Indeed, in addition to the text, a reviewer could include a photo of an expensive but inadequate dish or a dirty bath in luxury hotel. We set the value of c to $c = 0, 0.5, 1$ if (1) no image is included in the review (i.e., $I = \emptyset$), (2) $I \neq \emptyset$, (3) the content of text and images is recognized as reliable and coherent, respectively. Obviously, the case (3) (i.e., $c = 1$) requires semi-automatic activities and the intervention of a third party. Thus, this possibility is here only hypothesized but it is actually considered as a future issue (thus not included in the implementation).

Concerning the presence of witnesses (i.e., w), currently, no mechanism is provided by TripAdvisor. We thus propose to include a simple mechanism to cover this gap as follows. A user could indicate other people who can confirm his presence in an activity. The date d of reviews can be used to validate this declaration. We set $w = 0, 0.5, 1$ if (1) no witness exist, (2) there is just one witness, (3) there are multiple witnesses, respectively.

Concerning the confirmation by the operator p, no mechanism is at moment provided by TripAdvisor, so that we propose the inclusion of the following (no-cost) system. We expect that the operator issues a *ticket* to the reviewer, and the ticket is a credential needed to the reviewer to provide a review. To contrast the possibility that an operator carries out self-promoting attacks (by generating fake tickets), the idea is that the reviewer has a sequence of tickets, each generated by an operator who he has visited, but for reviewing the i-th operator, he has to use the ticket supplied from the $(i-1)$-th operator. The first ticket is generated from the system when creating the reviewer account. We set $p = 1$ if the review uses a valid ticket, $p = 0$ otherwise.

The value $trust(t_r) = h(c, w, p)$ is thus obtained by evaluating the function h on the values c, w, p. As function h we choose a linear combination of the three variables: $k_1 \cdot c + k_2 \cdot w + k_3 \cdot p$, where k_1, k_2 and k_3 are parameters of the system to tune through simulation and practical experience.

As a first step, we have to combine the value $trust(u)$ with the value $trust(t_r)$ so obtained by computing $trust(r_s) = f(trust(u), trust(t_r))$. Again, for the function f we choose a linear combination $k_3 \cdot trust(u) + k_4 \cdot trust(t_r)$.

Finally, the normalized review is computed as $r_s^* = \langle u, d, g(k), t, i \rangle$, where we define $g(k)$ as $g(k) = \frac{(\alpha + trust(r_s)) \cdot k}{\sum_{r_s' \in R(S)} (\alpha + trust(r_s'))}$, where α is s suitable (small) offset to avoid null terms. This way, the mean computed over all the scores obtained by the operator s is just the mean weighted by trust values (shifted by α) of each review.

5.4 Implementation Issues and Validation

In this section, we describe our strategy to build and validate the system implementing the approach presented in this chapter. We point out that, this is a work in progress carried out within industrial project. For this reason, we give here only some details about the strategy we are following for the validation of our proposal.

The data extraction strategy used to crawl TripAdvisor information leverages on HMTL parsing. Processing HTML to obtain social data is a low-level way of dealing with web data and requires much time because it needs to analyze all context information from the page source code. However, it remains the only solution because (i) TripAdvisor API content is not publicly available and is only for licensed partners and (ii) TripAdvisor APIs do not make available information about reviewers, indispensable to apply our strategy. After the extraction of data about reviewers, reviews and the corresponding hotels or restaurants (we restrict to these categories of operators), we stored them in a MongoDB server, a document-oriented database suitable for handling big data. Information stored in such a database is, then, processed to compute the parameters described in Section 5.2 by means of a Java prototype running on a personal computer equipped with a 4.0GHz Intel i7 CPU, 16 GB of RAM and Ubuntu 14.04.

Concerning the experimental validation campaign, the first issue to address is the computation of the parameter $trust(u)$, representing the level of assurance of identity proofing of the customer u who have authored the review. At the moment, TripAdvisor is not equipped with a token-based registration mechanism hence we cannot use this information. However, it is possible to access TripAdvisor by using an existing social network profile. Therefore, we partition users who have a standard TripAdvisor account and those accessing the system via a social network account. The supported social networks are Facebook and Google+ and, once a user accesses the system using an account of one of these social sites, a new TripAdvisor profile is built on the basis of the user information gathered from the social network.

As for the computation of the parameter $trust(t_r)$, which measures the trustworthiness of the transaction, we rely on two pieces of information obtained from TripAdvisor: pictures and witnesses. Concerning the former, although the functionality allowing the insertion of pictures about a TripAdvisor service providers has always been available on the system, pictures were originally not linked to any review. Only recently TripAdvisor has added the possibility of enriching a review with pictures of the visited place and, hence, now those pictures are directly associated with the specific review. Therefore, we can extract this information by parsing the HTML of the page containing a review. Listing 5.1 shows an example of such HTML code, in which Line 2 contains the review identifier, whereas the presence of the HTML tag `img` at Lines 16-17 means that there is a picture associated with this review. The attributes concerning the picture are listed in Lines 22- 25. Specifically, the field `data` (Line 25) is the `url` of the picture itself. The presence of a picture is used to compute the trustworthiness of the review as defined in Section 5.2.

As for the second information useful for the computation of $trust(t_r)$ (i.e., presence of witnesses), there is not a direct way to extract it from TripAdvisor even if, recently, some advanced functionalities leveraging on social network friendships have been introduced in the system. However, for the TripAdvisor account associated with a social network account (i.e., Facebook or Google+) we can combine TripAdvisor data with some knowledge derived from the involved social sites. In more detail, by using the publicly available system *SNAKE* [64], we can extract the friends of an account in a social network and, then, identify the witnesses for a review r posted by a TripAdvisor account created via a social network account a through the following algorithm:

1. Create the set R of the reviews having the same visiting date as r.
2. Create the set A_R of the social network accounts of TripAdvisor users who generated a review in R.
3. Extract the set F_a of social network friends of a by using the SNAKE system.
4. Compute $I = F_a \bigcap A_R$.

The output of the algorithm described above is the set I is obtained as the intersection between two sets of social network accounts, and represents the accounts of the friends of a (F_a) who posted a review about the same operator and in the same date as u_f. Observe that, the

```
<div class="entry">
<p property="v:description" id="review_252184***">
Very clean room [...]
</p>
</div>
<div class="media␣ug_thumb␣thumbCols1">
<div class="thumbnails">
<div class="photosInline">
<ul>
<li class="␣ThumbNailContainer">
<span class="u_/LocationPhotoDirectLink***"
onclick="ta.trackEventOnPage('Reviews','user_photo','image');
␣ta.util.cookie.setPIDCookie(5042);
␣return␣ta.call('ta.overlays.Factory.albumsLB',
␣event,␣this,␣6509342,␣274707,␣2,␣122220566);">
<img id='lazyload_45931090_1' height='50' alt='' width='50'
class='reviewInlineImage' src='http://e2.tacdn.com/***'/>
</span>
[...]
var lazyImgs = [
[...]
,{"id":"lazyload_45931090_1",
"tagType":"img","scroll":true,"priority":100,
"logerror":false,
"data":"http://media-cdn.tripadvisor.com/media/***"}
[...]
```

Listing 5.1: Structure of a review.

step 2 described above involves only friends from a single social network, for example Facebook. However, because nowadays most of users have accounts on multiple social networks, still referring to the example above, a more complete solution could take also Google+ contacts into account. To gather information about these additional accounts, the identification of a Google+ alter account of the user owning the account *a* (whether it exists) is required. A common approach to address this issue is typically based on profile matching techniques. This task could be carried out by using, for instance, the approach of [61]. In particular, to find matching profile, the string similarity between the associated usernames is combined with a contribution based on a notion of structural similarity among the common neighbors.

So far, we have described how to obtain from TripAdvisor data, the information required to compute the parameters introduced in Section 5.2. To validate the normalized scores computed by our approach, they should be compared with on the actual scores of the operators, that is the score obtained in case no malicious reviews are present in the system. Because obtaining these actual scores is not feasible, we adopt a sub-optimal strategy, which relies on the information available on another well-known travel Website, namely Booking.com [33]. Although the review mechanism of Booking.com could in principle be subject to some attacks, the additional features of this system (such as, the fact that reviews can be posted only by users who actually performed a reservation to the service provider) make it more robust w.r.t. TripAdvisor. In particular, according to our model, Booking.com fulfills the maximum level of assurance for both identity and transaction. Therefore, we are adopting data from Booking.com as ground-truth in our validation campaign. Moreover, to further validate the results of our approach, we can refer to the well-known *Wisdom of Crowd* principle of crowdsourcing mechanism [121]. Under this assumption, if a service provider receives, over time, a very high number of reviews, the score will be highly trustable due to the dejection of the noise caused by malicious outliers. For this reason, given an operator with a long history of reviews, we are testing our approach by considering a snapshot of TripAdvisor in a past date. By using this old information, we compute our normalized scores and compare our results with the score achieved by the operator in the present time.

Both the above validation activities are in progress so we cannot present conclusive results here although we are currently obtaining encouraging performances.

5.5 Related Work

Nowadays a great variety of computer applications are decentralized in open distributed systems. Moreover they belong to widely differing areas (such as peer-to-peer computing, semantic Web, Web services, autonomic computing, and so on), and are characterized by the fact that their components are autonomous actors, spread throughout a network and cooperating in order to achieve the design objectives.

In most of these systems, however, there is a need to apply trust and reputation-based approaches to improve their effectiveness. For this reason, in recent years, a lot of models have been formalized due to the increasing recognition of their roles in controlling social order in open systems [194, 144].

To provide trust metrics, these models take advantage of different information sources. In general, these sources can be divided into two groups: (i) direct interactions [144, 206] and (ii) feedbacks provided by other members of the society about experiences they had in the past [194, 131]. Specifically, the model FIRE, described in [126], formalizes suitable concepts of trust and reputation for open multi-agent systems. This approach is based on different metrics such as, interaction trust, role-based trust, witness reputation and certified reputation. The model presented in [206] takes advantage of social relations between agents, whereas, in [144] the hybrid trust model T-REX is formalized. It combines witness reputation and actors' personal experience as well as it integrates multiagent systems (MAS) with Semantic Web technology.

The AgentTMS model proposed in [194] is based on social relationships among agents and, in particular, on types and strengths of these interactions. [131] proposes a reputation model for trust management in a semantic P2P Grid using fuzzy theory to compute a peer trust level. The authors of TRAVOS [227] develop a trust metrics which uses probability theory and past interactions between agents. In case of lack of personal experience between agents, the model draws upon reputation information gathered from third parties.

All the approaches presented above apply on open Multi-Agent Systems (MAS) and it is precisely in this context that the notion of *certified reputation* has been firstly introduced [123, 125]. Specifically, the Certified Reputation of an agent is the reputation that is derived from third-party references about its previous performance consisting of a number of certified references about its behavior on particular tasks. Such information is obtained and stored by the agent itself and made available to any other agent that wishes to evaluate its trustworthiness for further interactions.

Our work gives a contribution in this field, by proposing a reputation model based on service providers, users and feedbacks, implementing the theoretical notion of certified reputation for the TripAdvisor travel service. Our goal is to define a strategy to *normalize* feedback scores towards reliable values.

Part II

Private and Public Physical Organizations

This second part focuses on strategies assessing privacy and security on both public and private physical organizations.

The first three proposed approaches lie in the field of e-governement, that is the use of electronic communications devices, computers and the Internet to provide public services to citizens. One of the main problems in this context is the authentication of transactions. In order to solve this issue we proposed in Chapter 6 a new electronic signature protocol not using public-key encryption, qualified signature creation devices, or qualified certificates. This protocol relies on an innovative model, which exploits the power of online social networks of sharing information by spreading out on them the signature functions. Always in the context of e-government, another critical aspect is authentication of users, especially in scenarios where user's privacy is threatened by *honest-but-curious* cloud providers. In Chapter 7 we propose an authentication scheme supporting full anonymity of users and unlinkability of service requests. This is done by combining a multi-party cryptographic protocol with the use of a cooperative P2P-based approach to access services in the cloud.

Another interesting and new aspect we focus on is related to Blockchain technology, which allows mutually distrustful parties to transact safely without trusted third parties and avoiding high legal and transactional cost. In Chapter 8, we propose an alternative public ledger that, instead of the P2P network and the protocol of Blockchain, leverages the popular social network Twitter, and builds a meshed chain of tweets to ensure transaction security.

Then, our focus moves towards some approaches in the field of urban security. More and more often, both private and public organizations rely on cloud-based services supplied by third parties. This happens in many cases, for those companies that do not choose to implement private clouds and for those governments in which national clouds are not adopted. Urban security is one of the settings in which monitoring systems produce a huge amount of data. An emblematic case is that of video surveillance, especially if a capillary solution is adopted. Therefore, it is plausible that the municipality finds convenient to move video surveillance data to a third-party cloud. In this case, the trustworthiness of the cloud provider becomes a critical point. Among other threats, the possibility that the cloud does not return intact responses may have severe consequences. In order to solve this issue, Chapter 9 proposes an approach to allow users to verify that cloud query results are *complete* (i.e., no qualifying tuples are omitted), *fresh* (i.e., the newest version of the results are returned), and *correct* (i.e., the result values are not corrupted).

Still in the context of urban security, another aspect concerns the trade-off between people privacy and the need to guarantee security standards in critical environments also by adopting continuous monitoring. To solve this trade-off, we propose a complex system that borrows the concept of k-anonymity from databases and applies it to build a new approach allowing the identification of people accessing a zone providing a desired level of privacy. Two interesting application scenarios are presented in Chapter 10 and in Chapter 11, respectively Critical infrastructures and for Assistive Living Facilities.

6

Security of Transactions in E-government: A Social-Network-Based Advanced Electronic Signature

In e-government applications, the use of qualified electronic signature will always be increasing in the next future. However, there are some aspects related to costs and usability that limit its diffusion. This makes timely and important the issue of designing new signature protocols that relax the heaviest features of public-key-encryption based qualified electronic signature, yet keeping the characteristics of advanced electronic signature. We propose a new electronic signature protocol that does not use public-key encryption, qualified signature creation devices, or qualified certificates. The protocol relies on an innovative model, which exploits the power of OSNs of sharing information by spreading out on them the signature functions. A deep security analysis shows that the proposed approach is robust against known attacks on digital signatures and an implementation using `Twitter` *services makes it realistic and effective.*

6.1 Background

In this section, we introduce some background notions concerning the different kinds of electronic signature allowed by the current European regulations. Indeed, the solution proposed in this chapter aims at being framed in the European regulatory framework to be concrete and effective. As observed in the Introduction, the reference to the European regulatory framework does not limit the generality of our proposal, as the US regulatory framework provides a broader notion of electronic signature.

An *advanced electronic signature* [1] is defined as an electronic signature that meets the following requirements: (1) it is uniquely linked to the signatory, (2) it is capable of identifying the signatory, (3) it is created using means that the signatory can maintain under his sole control, and (4) it is linked to the data to which it relates in such a manner that any subsequent change of the data is detectable. An advanced electronic signature satisfying certain security and quality requirements is said *qualified electronic signatures*. In particular, [1] states that a qualified electronic signature shall: (1) be an advanced electronic signature, (2) be based on a qualified certificate issued by a suitably certified certification service provider (as specified in the Annex I of [1], and (3) be created by a secure signature creation device that meets specific functional conditions which are also laid down in [1]. The main benefit of qualified electronic signature is that they can benefit from an automatic legal equivalence to handwritten signatures across the European Union.

Despite these signatures hold the principles of technological neutrality, the *de-facto* standard for the electronic signature with automatic legal value is public-key-encryption-based qualified electronic signature, which enforces the utilization of qualified PKI certificates and qualified signature creation devices. Even the existing solutions of advanced electronic signature makes use of public-key encryption. Usually it is obtained by relaxing some security requirement of qualified electronic signature, or by combining the use of dynamic signature (capturing the biometric features of an hand-written signature) with public-key encryption to link the final signature also to the content of the document.

We assume that the reader knows how a signature based on public-key encryption (typically RSA [201]) works. A signed document produces what it is usually called *cryptographic message*, which can be encoded in several formats, such as PKCS#7, CMS, CAdES, XAdES or pdf.

One of the drawbacks of qualified electronic signature is the need of having a secure signature creation device to sign a document, which increases the cost and decrease the usability of the solution. For this reason, beside smart-card-based signatures, a remote digital signature has been introduced. The remote signature is a type of signature, accessible via network (Intranet and/or Internet), in which the signatory's private key is stored along with the signature certificate, within a secure remote server (based on a HSM - Hardware Security Module) by an accredited provider. HSMs are devices used for data encryption and decryption and host one or more cryptographic keys that respond to automated or manual commands. They safeguards and manages digital keys and have the capability to detect an attack on their surface and securely delete the sensitive content stored in their memory. In particular, HSMs are specific hardware designed to protect key against any kind of logical and physical manumission or extraction of cryptographic material from its environment.

The HSMs are normally hardware that passed by certification procedure. The most important are FIPS 140-2, a certification involved by USA's Department of Commerce, and Common Criteria [11], developed by a consortium that wants to obtain protection profiles for such equipment. Key management life cycle has been studied by many researchers [187]. In particular, Menezes et al. [173] discuss the public key management in a general context, considering user registration and initialization to key revocation. Note that an electronic signature solution asserts that all signatories are originally identified by the signature master with a given level of assurance. In particular, the level of assurance of authentication in case of remote signature (using HSMs) is at least the (multi-factor) level of assurance 3 of the NIST model.

Another important point is the level of assurance required for the PKI registration phase. In this phase, the owner of the signature certificate is identified and registered by the certification service provider. As for authentication, the level of assurance of the identification can be referred to the NIST framework [66]. For qualified certificates, at least level 3 in-person is required, i.e., the possession of verified current primary government picture ID that contains applicant picture and either address of record or nationality of record (e.g., drivers license or passport).

6.2 The Social-Network-Based Signature Model

In this section, we present the abstract model underlying our proposal. In the sequel of the chapter, we instantiate this model to a concrete scenario involving real-life entities and addressing all technical issues. The second step will be a prototype implementation of the concrete instantiation of the model. The model is composed of:

The Actors: The entities involved in the signature process: The *signature master* M, the *signatory* S, and the *social network* T.

A Posting Integrity Mechanism: A mechanism provided to the signatories and the signature master to prevent the corruption of their posting timeline.

A Registration Protocol: The initial phase, where the real-life identity of signatories is associated with a profile in the social network T and a secure sharing of this information is established.

A Signature Generation Protocol: The protocol followed by signatories to sign a document.

A Revocation Protocol: The protocol aimed at inhibiting a given signatory to validly run the signature protocol from the moment of revocation on.

A Signature Verification Protocol: The protocol followed by any recipient of a signed document to verify the validity of the signature.

The Security Parameters: The parameters concurring at establishing the level of security of the whole signature process. Now, we describe in detail each of the above component.

Actors. Any instance of the model consists of one signature master M, a set of signatories (that can be dynamically updated), and one social network T. Only T is assumed to be trusted. Both signatories and signature master have a profile in T. We assume that this social network supports (at least) (1) a friendship relation, (2) the possibility for users to post a text message equipped with a timestamp in a public section of their profile, (3) the possibility of searching for a message posted in a public section by means of usual text-search tools. For each signatory S, a friendship with the signature master is established in the social network T. We require that in T , automatically, any message posted in his public section by a friend S of M is conditionally (according to a given condition set by M) replicated by M in its public section, by including the same information, in such a way that the deletion of the original message does not propagate to the replicated messages.

Posting Integrity Mechanism. Each signatory S is provided with a second preimage resistant trap-door function $f : \mathcal{D} \rightarrow \mathcal{C}$, where $\mathcal{D}$ and $\mathcal{C}$ are finite, and a secret information p such that, for a given output y of the function f, finding x such that $y = f(x)$ is feasible only for S, who owns p. Each post (belonging to the signature protocol) of a signatory S is associated with a number y such that the next post of the same signatory is associated with a number x such that $y = f(x)$. Consider that only S is able to generate a legal new post because he is the only party able to compute the number to associate with the post. As shown in Section 6.3, this measure prevents attacks based on the posting-timeline corruption.

Registration Protocol. All (real-life) signatories are originally identified (and assigned to a social network profile) by the signature master. The first step of the registration protocol is the publication done by the signatory S in the public section of his profile of the first message

$\langle y_0, I, ID_S \rangle$, where y_0 is obtained by the posting integrity mechanism above, I is an information confirming the association of S with the signature master M, and ID_S is defined above. For each signatory S, the association between a real-life identity of S and his social network profile is published as welcome message for S $\langle y_0, J, ID_S \rangle$ in the public section of the signature master profile, where y_0 is the same as the registration message, J is an information describing the real-life identity of S, and ID_S is the identifier of S in the social network of T.

Signature Generation Protocol. Let D_i be the i-th ($i > 0$) document (in order of time) being signed by the signatory S, ID_S be the identifier of S in the social network of T, and h be a k-bit cryptographic hash function. We define the *signature message* as $\langle y_i, h(D_i), ID_S \rangle$, where y_i is a number. A signature message $\langle y_i, h(D_i), ID_S \rangle$ is said *linked (in S)* if (recursively) $f(y_i) = y_{i-1}$ and either $\langle y_{i-1}, h(D_{i-1}), ID_S \rangle$ is the latest linked signature message associated with ID_S occurring in T or $\langle y_{i-1}, I, ID_S \rangle$ is the first message posted by S at the registration phase.

The signature generation protocol consists of the publication done by S in the public section of his profile of a *signature message* $\langle y_i, h(D_i), ID_S \rangle$, which enforces that M either:

1. replicates this message in its public section (thus producing a message also called *confirmation message*), if it is linked in S and there not exists another message $\langle y_i, h(D^*), ID_S \rangle$ in S or M with $h(D_i) \neq h(D^*)$, or
2. publishes an *aborting message (for y_i)* $\langle y_i, h(D_i), ID_S \rangle$ (meaning that the signature procedure fails), if the signature message is linked in S and there exists another message $\langle y_i, h(D^*), ID_S \rangle$ in S with $h(D_i) \neq h(D^*)$, or
3. does nothing, otherwise.

Moreover, the protocol enforces that the maximum time between the publication of the message by S and the publication of the corresponding message by M is not greater than Δt, which we call *reaction time constraint*. The security mechanism based on the trap-door results in a chain of signature messages starting from the original registration message $\langle y_0, I, ID_S \rangle$ that only the signatory is able to generate. This chain is also published by the signature master M due to the replication mechanism described above. According to the protocol, the chain is replicated by M for all linked signature messages through confirmation/aborting messages. However, if a signature message is not linked, no message is published by M. In other words, no confirmation message for a signature message $\langle y_i, h(D_i), ID_S \rangle$ such that $f(y_i) \neq y_{i-1}$ may occur in the posting timeline of M or such that the iterated self composition of f to y_i does not allow us to reach y_0 after i steps, where $\langle y_0, I, ID_S \rangle$ is the first message posted by S at the registration phase.

As we are interested in managing also states not generated by the signature generation protocol (as for example those arising from the deletion of messages from the timelines of S and M), we need to introduce a *stronger* notion of linkness. A signature message $\langle y_i, h(D_i), ID_S \rangle$ is said *(k-)linked (in S)* if there exists $k > 0$ such that a message $\langle f^k(y), h(D_k), ID_S \rangle$ occurs in the public section of S and either its confirmation or a corresponding aborting message appears in the public section of M, and for each $0 < j < k$ the message $\langle f^j(y), h(D_j), ID_S \rangle$ belongs to S or its confirmation (or the aborting message) belongs to M, where f^k denotes the self composition

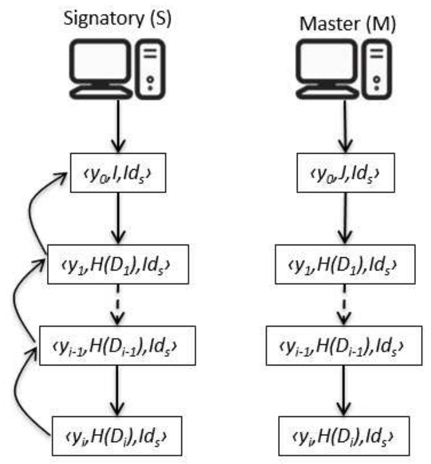


Fig. 6.1: An example of a linked message.

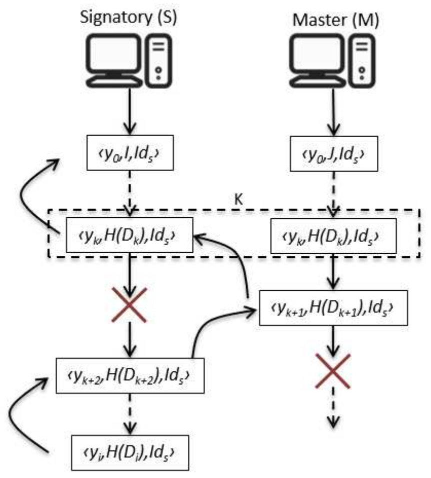


Fig. 6.2: An example of a k-linked message.

of the function f for k times[1]. Given a message $\langle y, h(D), ID_S\rangle$, denoting by n the value such that $\langle f^n(y), I, ID_S\rangle$ is the first message posted by S at the registration phase, it is easy to see that if the message is k-linked (for $0 \le k \le n$), it is j-linked too, for any $k \le j \le n$. Moreover, it is also linked. This immediately follows from the definition of the signature generation protocol. Therefore, to prove that a message is linked, it suffices to find the minimum k such that the message is k-linked, thus in general saving computation. In Figures 6.1 and 6.2, we report some examples of timelines to better explain the above notions.

Finally, Algorithm 2 schematizes the above protocol.

[1] Observe that no computability problems may arise (in the worst case of a negative instance) as f operates on finite sets. Anyway, to avoid infeasible computation in a concrete implementation it suffices to bound the maximum number of possible signature messages (per signatory) by a value u, so that k, if any, is less or equal to u. Moreover, as f is efficient, u can be large enough to cover all possible signatures of a user.

Algorithm 2 Signature Generation Protocol

```
Notation  $S$: the document signatory
Notation  $M$: the signature master
Notation  $T$: a social network
Notation  $h$: a $k$-bit cryptographic hash function
Input  $ID_S$: the identifier of $S$ in $T$
Input  $D_i$: the $i$-th document being signed by $S$
Input  $y_i$: a number
1: $S$ publishes $\langle y_i, h(D_i), ID_S \rangle$
2: if ($\langle y_i, h(D_i), ID_S \rangle$ is linked in $S$) then
3:     if ($\nexists \langle y_i, h(D^*), ID_S \rangle$ in $S$ and $\nexists \langle y_i, h(D^*), ID_S \rangle$ in $M$) then
4:         $M$ publishes $\langle y_i, h(D_i), ID_S \rangle$
5:     else if (($h(D^*) \neq h(D_i)$)) then
6:         $M$ publishes the aborting message $\langle y_i, h(D_i), ID_S \rangle$ in $\Delta t$
7:     end if
8: end if
```

Revocation protocol. The revocation of a signatory S is done by the signature master by simply removing the friendship of S and by publishing a revocation message identifying time and reasons of revocation.

Signature Verification Protocol. Concerning the validity of the signature, our verification procedure, for any detected potential signatory, outputs two possible values each equipped with an attribute, resulting in four different outcomes `(value, attribute)`: (1) `(valid, safe)`, (2) `(invalid, safe)`, (3) `(valid, unsafe)`, and (4) `(invalid, unsafe)`. The full meaning of attributes will be clear in Section 6.3. Basically, a `safe` outcome means that no anomaly is detected. Conversely, an `unsafe` outcome indicates the occurrence of an anomaly (i.e., attack or failure). Interestingly, due to the value returned, among `valid` and `invalid`, the anomaly does not affect the (even legal) effects of the signature. Therefore, the `attribute` is an extra information given as warning to detect an anomaly, as for example a blocked attempt of attack. As it will be clear in Section 6.3.2, besides the `attribute`, the signature verification procedure could return in principle further extra information related to the origin of the anomaly. For simplicity, we do not consider this feature here.

Let D be the document whose signature has to be verified. The protocol works as follow. First, the digest $h(D)$ is computed. Then, $h(D)$ is searched among the public information posted in the social network T. If $h(D)$ is not found the verification returns the outcome `(invalid, safe)`, with no other information. In other words, the document D is detected as not signed. Otherwise, we have the following cases:

1. Both signature message $\langle y, h(D), ID_S \rangle$ and its corresponding confirmation message exist and S is not revoked. In this case, the verification procedure returns `(valid, safe)` w.r.t S. Observe that, according to the signature generation protocol, the presence of the confirmation message ensures that $\langle y, h(D), ID_S \rangle$ is linked in S.
2. A signature message $\langle y, h(D), ID_S \rangle$ exists either in S or M (not in both). In this case, the verification procedure returns:
   a) `(valid, unsafe)` w.r.t S if either:

i. the found message $\langle y, h(D), ID_S \rangle$ is linked in S, there not exists an aborting message for y in M with time delay w.r.t. the signature message less than (or equal to) Δt (thus compliant with the reaction time constraint Δt), and S is not revoked, or

ii. the found message $\langle y, h(D), ID_S \rangle$ is linked in S, a message exists (in S or M) $\langle y, h(D'), ID_S \rangle$ such that $D \neq D'$, the message with lowest timestamp between $\langle y, h(D), ID_S \rangle$ and $\langle y, h(D'), ID_S \rangle$ contains $h(D)$, and S is not revoked,

b) `(invalid, unsafe)` w.r.t S, otherwise.

Note that in the multiple signatories the signature verification protocol associates each signatory with an outcome `(value, attribute)`. An algorithm describing this protocol is shown in Algorithm 3.

Algorithm 3 Signature Verification Protocol

```
Notation  $T$: a social network
Notation  $S$: the document signatory
Notation  $M$: the signature master
Notation  $h$: a $k$-bit cryptographic hash function
Notation  $\Delta t$: the reaction time constraint
Notation  $ID_S$: the identifier of $S$ in $T$
Notation  $m$: a signature message $\langle y, h(D), ID_S \rangle$ for the document D
Notation  $m'$: a signature message $\langle y, h(D'), ID_S \rangle$ $(D' \neq D)$
Notation  $\overline{m}$: an aborting message $\langle -y, h(D), ID_S \rangle$ on $y$
Notation  $t_m$: time delay w.r.t. the signature message $m$
Notation  $min(m, m')$: a function returning the message with the lowest timestamp
Input  $D$: the document to be verified
Output  $results$: an empty list
 1: Compute $h(D)$;
 2: Search $\langle h(D) \rangle$ on $T$;
 3: if ($\nexists \langle h(D) \rangle$) then
 4:     add $\langle$null,(invalid,safe)$\rangle$ to $results$;
 5:     return $results$;
 6: end if
 7: for each $ID_S$ found do
 8:     if ($\exists$ $m$ in $S$ and $\exists$ $m$ in $M$ and $S$ is not revoked then
 9:         add $\langle ID_S$,(valid,safe)$\rangle$ to $results$;
10:     else
11:         if ($\exists$ $m$ in $S$ or $\exists$ $m$ in $M$) then
12:             if ($m$ is linked in $S$ and $\nexists$ $\overline{m}$ in $M$ with $t_m \leq \Delta t$ and $S$ is not revoked) then
13:                 add $\langle ID_S$,(valid,unsafe)$\rangle$ to $results$;
14:             else if ($m$ is linked in $S$ and ($\exists$ $m'$ in $S$ or $\exists$ $m'$ in $M$) and $min(m, m') = m$ and $S$ is not revoked) then
15:                 add $\langle ID_S$,(invalid,safe)$\rangle$ to $results$;
16:             else
17:                 add $\langle ID_S$,(invalid,unsafe)$\rangle$ to $results$;
18:             end if
19:         else
20:             add $\langle ID_S$,(invalid,unsafe)$\rangle$ to $results$;
21:         end if
22:     end if
23: end for
24: return $results$;
```

Security Parameters. The level of assurance of the authentication procedure of users when logging into their social network profile, say $LoA(Auth)$, is a parameter of the model (the value of this parameter can be assumed belonging to the score given by NIST in [66], thus from 1 to 4). According to our model, an electronic signature solution enforces that all (real-life) signatories are originally identified (and assigned to a social network profile) by the signature master (registration protocol) with a given level of assurance, say $\langle LoA(Id), p\rangle$, which is a pair of parameters denoting with $LoA(Id)$ the identification level of assurance adopted in the registration phase (among the levels given by NIST in [66], thus from 1 to 4), and with p if the identification is *in-person* ($p = 1$) or *remote* ($p = 0$). $\langle LoA(Id), p\rangle$ induces the level of assurance of the *identification* function of the electronic signature. $LoA(Auth)$, $\langle LoA(Id), p\rangle$, and the other security features of a concrete instantiation of the model induces the level of assurance of the electronic signature w.r.t. both *indicative* and *declaratory* functions of the signature [221], thus the security of the non-repudiation service implemented by the signature itself. Other parameters could exist, depending on the concrete instantiation of the model.

6.3 Security Model

In this section, we state the security properties required for the proposed e-signature protocol. We denote by SM the security model so obtained. We remark that we consider attacks identified on the basis of the taxonomy proposed in [118], from most enabled adversaries who are "enabled insiders", namely, internal or external users and the signature master M. Hence, a protocol which is secure against such adversaries is also secure against any other adversary who is an outsider or a normal user in the system. Observe that even though most of the anomalies based on software/network (fault-based) failures are subsumed by the considered attacks, the aim of this section is to analyze the security of our protocol, not its dependability in the largest meaning.

6.3.1 Assumptions, Security Properties, and Attacks

Our threat model realistically considers the following assumptions:

A1 A signature system is secure w.r.t. a given attack if it is (at least) as secure as CADES/X-AdES qualified electronic signature system compliant to [1].

A2 Collision, preimage and second preimage attacks on the cryptographic hash function h are infeasible;

A3 The social network T acts as a trusted third party;

A4 The attacker cannot add or compromise information shown on the social network accounts of the signature master and signatories with no legal authentication;

A5 The authentication on the social network is configured to require a double-factor authentication (level of assurance 3 of the NIST model [66]);

A6 The initial registration is done at level of assurance 3 in-person of the NIST model [66];

A7 The occurrence of the initial registration is provable by both parties (the master and the signatory), as well as the integrity of the published registration message;

A8 The exact duration of the signatory's status granting the right to use the signature service can be proven by means of secure information external to the system (e.g., documents kept by the master);

A9 The secret enabling the trap-door function of the posting integrity mechanism is managed with a level of security at least equal to that of CADES/XAdES qualified electronic signature systems.

A10 No collusion may occur between master and users.

Concerning Assumptions **A1**, according to current European rules [1], we observe that the level of security of the management of secrets in a CADES/XAdES qualified electronic signature is bounded by how users are authenticated to access HSMs (Hardware Security Modules) services (considering that attacks on HSMs are definitely more difficult and there is no proof of concept that the known vulnerabilities can be exploited to jeopardize the signature process). In the case of HSM, users' private keys are kept by a TTP (the HSM provider), and their utilization (to generate signatures) is enabled by the signatory by means of a double-factor authentication (level of assurance 3 of the NIST model [66]). This prevent the success of keylogger-based attacks. So we consider this as lower bound security for how we manage the secrets of our protocol. Therefore, Assumption **A9** means that the adopted solution is able to protect the secret against at least a keylogger-based attack on the device client side. Now, we are ready to state the security properties of our protocol. Thanks to Assumption **A10**, we consider only the cases where the attacker is either a signatory or the signature master. Observe that, according to Assumption **A1**, our aim is to assess that our solution can be classified as advanced electronic signature, as defined in [1].

Security Property 1 (SP1) - ***Document Authenticity.*** SP1 is defined as follows: *A signed document should be always verified as issued by the real signatory. In other words, the signature should be always a proof of origin.*

The attack model we consider to describe how this property can be threaten is the following:

Attack AA1: An adversary tries to use a fake Twitter profile to jeopardize the proof of origin of a social signature of a legal user.

Attack AA2: An adversary attempts to impersonate another identity by deceiving the registration phase.

Attack AA3: An adversary attempts to impersonate another identity by stealing everything the legal user needs to sign a document (by social engineering, interception, observation, endpoint compromise, guessing, of signature creation data).

Attack AA4: The attacker substitutes the original signature by a signature generated by himself on the same document, compromising the proof of origin.

Attack AA5: The attacker tries to attribute the signature of a document to a victim user.

Security Property 2 - ***Document Integrity*** SP2 (Security Property 2) is defined as follows: *The binary content of a signed document should be always verified as exactly equal to the original document.*

The attack model we consider to describe how this property can be threaten is the following:

Attack AI1: An adversary tries to forge a valid signature starting from a signature of the victim (i.e., selective forgery attack).

Attack AI2: An adversary tries to forge a pair of documents with same digest and submit one of the two documents to the victim to be signed (i.e., existential forgery attack).

Attack AI3: An adversary tries to forge a new document and, accordingly, the associated social signature of a victim user (i.e., existential forgery attack).

Security Property 3 (SP3) - ***Signature Verification Dependability*** SP3 is defined as follows: *The signature validity verification should be always dependable.*

The attack model we consider to describe how this property can be threaten is the following:

Attack AD1: An adversary tries to repudiate its Twitter account or its use, thus invalidating the signature validity verification.

Attack AD2: An adversary tries to repudiate a social signature by illegal operations (w.r.t. the signature generation protocol) on his timeline.

Attack AD3: An adversary tries to repudiate a social signature by jeopardizing the revocation system.

Attack AD4: An external adversary (different from the signatory) tries to make a valid signature generated by a certain user be verified as invalid.

Attack AD5: An adversary tries to make a signature generated over a fraudulently modified document be verified as valid.

Attack AD6: The master, playing as adversary, tries to make invalid a valid signature verification by illegal operations (w.r.t. the signature generation protocol) on his timeline.

Security Property 4 (SP4) - ***Document Immutability*** SP4 is defined as follows: *The presentation of a signed document (i.e., what the document shows) should be always the same, not dependent on external variables such as time, host device, etc.*

Attack AP1: An adversary includes dynamic contents in a document being signed.

Attack AP2: An adversary corrupts the fonts of the host system.

Attack AP3: An adversary produces a polymorphic file [36] as document being signed.

6.3.2 Security Analysis

In this section, we analyze the practical security (as stated by Assumption **A1**) of our signature system. We consider separately all the security properties we have to guarantee, which are document authenticity, document integrity, signature verification dependability, and document immutability, as stated in the security model presented in the previous section. Even though multiple signatories of a given document might occur, our security analysis focuses only on the case of a single signatory, without loss of generality. Indeed, it is easy to realize that user collusion cannot give any advantage. The following theorem states that the protocol satisfies the property SP1 of the security model SM.

Theorem 6.1 (Document Authenticity). *The protocol is secure against attacks AA1, AA2, AA3, AA4, and AA5.*

Proof. *Resistance to Attack AA1.* Recall that AA1 occurs whenever an adversary tries to use a fake social network profile to jeopardize the proof of origin of a social signature of a legal user. In particular, the attacker creates a profile with the victim's real-life information, and then tries to impersonate the victim. Let $\langle y, h(D), ID_S \rangle$ be a fake signature message whose aim is to attribute the signature of the document D to the signatory S. This attack is vanished by the fact that the signature generation protocol requires that the signature message is linked, to respond the value `valid`. But, thanks to Assumption **A7**, this cannot happen.
Resistance to Attack AA2. Recall that AA2 occurs whenever an adversary attempts to impersonate another identity by deceiving the registration phase. The security of our system against this attack is (at least) the same of CADES/XAdES qualified electronic signature systems, as the level of assurance required for the initial registration is the same (according to Assumption **A6**) and the master plays as third trusted party (according to Assumption **A3**).
Resistance to Attack AA3. Recall that AA3 occurs whenever an adversary attempts to impersonate another identity by stealing everything the legal user needs to sign a document (by social engineering, interception, observation, endpoint compromise, guessing of signature creation data). The security of our system against this attack is (at least) the same of CADES/XAdES qualified electronic signature systems, as the level of assurance of authentication required for that signature in case of remote signature (using HSMs) is the same required for our system (according to Assumption **A5**), that is level of assurance 3 of the NIST model [66].
Resistance to Attack AA4. Recall that this attack occurs whenever the attacker substitutes the original signature by a signature generated by himself on the same document, trying to compromise the proof of origin. The only way to to this for the attacker is to remove the original signature message from the victim's timeline and the confirmation message from master timeline and to perform a new signature on his social network profile. However, our system is resistant to this attack, according to Assumptions **A4** and **A5**.
Resistance to Attack AA5. Recall that this attack occurs whenever the attacker tries to attribute the signature of a document to a victim user B. This attack can be performed in three modes:

1. In the first mode the master plays the role of attacker. In this case, the master forges a message $\langle y, h(D), ID_S \rangle$, thus trying to attribute the signature of D to S. The signature verification protocol falls into the case 2(a)i, since $\langle y, h(D), ID_S \rangle$ cannot be found in S. But, to return a valid response, it requires that the message is linked in S. Since, for the message to be linked, y must be the preimage of y_i (where y_i belongs to the latest signature message of S) of the trap-door function whose secret is kept only by S. According to Assumption **A9**, our protocol is secure against this attack.
2. Also in the second mode the master plays the role of attacker. Suppose that a user A generates a linked signature message $\langle y, h(D), ID_A \rangle$ to sign the document D. According to

the signature generation protocol (see item 1), the master should replicates this message in its timeline. Instead to do this, the master posts the message $\langle y, h(D), ID_B \rangle$, thus trying to attribute the signature of D to B. The signature verification protocol falls into the case 2, since $\langle y, h(D), ID_B \rangle$ cannot be found in B. But, to return a valid response, it requires that the message is linked in B. This cannot happen, because y belongs to the chain of A, thus it is linked in A but not in B. Therefore, the signature verification protocol falls (1) into the case 2b, thus returning `(invalid, unsafe)` w.r.t. B, and (2) into the case 2(a)i, thus returning `(valid, unsafe)` w.r.t. A.

3. In the third mode the attacker is a user A. He just posts a linked (in A) signature message $\langle y, h(D), ID_B \rangle$, thus trying to attribute the signature of D to B. According to the signature generation protocol, the master does not publish the confirmation message because the signature message is linked in A but not in B. Moreover, the signature verification protocol falls into the case 2b, because the signature message is not linked in B, thus returning `(invalid, unsafe)` w.r.t. B. The attack is then contrasted.

Resistance to Attack AA6. Recall that this attack occurs whenever the master (playing here the role of attacker) tries to simulate the signature of a document by one of its employees. To do this, the master can proceed in two modes (naive and enhanced).
(1) In the naive mode, the master posts in its timeline a fresh message $\langle y, h(D), ID_S \rangle$ such that it is linked in S. If the master is able to do this, the attack succeeds, because the signature verification protocol would return `(valid, unsafe )`, according to item 2(a)i (i.e., simulating that S has deleted the signature message to repudiate the signature itself). However, this attack would require the knowledge of the secret owned by S, in order to compute y, i.e., the preimage of y_i, (where y_i belongs to the latest signature message of S) of the trap-door function. According to Assumption **A9**, our protocol is secure against this attack.
(2) In the enhanced mode, the master deletes (or does not publish) a confirmation message, say $\langle y, h(D), ID_S \rangle$ and forges a fake confirmation message, say $\langle y, h(D'), ID_S \rangle$, where $D' \neq D$, to falsely attribute the signature of the document D' to S. Indeed, in this case, $\langle y, h(D'), ID_S \rangle$ is linked in S. However, the timestamp of the forged confirmation message is more recent than the message $\langle y, h(D), ID_S \rangle$ occurring in the timeline of S. Thus, according to item 2(a)ii of the signature verification protocol, the verification procedure outputs `(valid, unsafe )` on D and `(invalid, unsafe )` on D'. Thus, the attack fails on both documents.

Now, by means of the following theorem, we state that the protocol satisfies the property SP2 of the security model SM.

Theorem 6.2 (Document Integrity). *The protocol is secure against attacks AI1, AI2, and AI3.*

Proof. *Resistance to Attack AI1.* Recall that this attack occurs whenever an adversary tries to forge a valid signature starting from a signature of the victim (i.e., selective forgery attack). It is easy to see that, thanks to Assumption **A7**, only the master could try this attack, because a valid signature message is always linked in the legal signatory. Suppose now the attacker is the master. In this case, it tries to forge a linked signature message $\langle y', h(D'), ID_S \rangle$ starting from a linked signature message $\langle y, h(D), ID_S \rangle$ published by the victim S. The aim of the

attack is to attribute to S the signature of the document D'. To do this, the attacker should be able to perform a second preimage attack on the trap-door function f, by finding y' such that $f(y) = f(y')$, which is infeasible for how f is chosen.
Resistance to Attack AI2. Recall that this attack occurs whenever an adversary tries to forge a pair of documents with same digest and submit one the two documents to the victim to be signed (i.e., existential forgery attack). The complexity of this attack, in our scheme, is that of a collision attack (i.e., birthday attack), on the k-bit cryptographic hash function h, thus $O(2^{k/2})$. Again, this attack is impracticable according to Assumption **A2**.
Resistance to Attack AI3. Recall that this attack occurs whenever an adversary tries to forge a new document and, accordingly, the associated social signature of a victim user (i.e., existential forgery attack). As the binary output of a signature creation is just the digest of the document an existential forgery attack in strict sense can be always successfully done. However, this digest plays the role of signature only if it is issued as signature message by the signatory (and the rest of the protocol is triggered). As a consequence the practical security against this attack is guaranteed by Theorem 6.1.

The following theorem states that the protocol satisfies the property SP3 of the security model SM.

Theorem 6.3 (Signature Verification Dependability). *The protocol is secure against attacks AD1, AD2, AD3, AD4, AD5, and AD6.*

Proof. *Resistance to Attack AD1*. Recall that this attack occurs whenever an adversary tries to repudiate its social network account or its use, thus invalidating the signature validity verification. The repudiation of the social network account can be done only by questioning about the security of the registration phase. According to Assumptions **A6** and **A1**, our system is secure against this attack. The repudiation of the social-network-account use can be done only by claiming the violation of the account. However, due to Assumptions **A4**, **A5**, and **A1**, we can conclude that our system is secure against this attack.
Resistance to Attack AD2. This attack may be performed by the user in three different modes.
(1) The first mode occurs whenever an adversary tries to repudiate a social signature by deleting the corresponding signature message by his timeline. According to our protocol, whenever the signatory issues a signature message in his social network account, the master generates the confirmation message on his social network page provided that the signature message is linked, by including the same information as the signature message. According to Assumptions **A4** and **A10**, the signature verification protocol will output `(valid, unsafe)`, because we fall into the case 2(a)i. Therefore, according to Assumptions **A5** and **A1** our system is secure against this attack.
(2) The second mode occurs whenever an adversary tries to repudiate a social signature by generating a signature message not linked, that is a message $\langle y_i, h(D_i), ID_S \rangle$ such that the iterated self composition of f to y_i does not allow us to reach y_0, where $\langle y_0, I, ID_S \rangle$ is the first message posted by S at the registration phase. The aim of the attacker is to trigger the confirmation message publication by the master. Indeed, according to 1 of the signature verification protocol, in this case the output of the verification protocol would be `(valid, safe)`, so a possibly

obligation about the document D_i would seem correctly satisfied to any recipient. In a second step, the plan of the attacker is to delete the message $\langle y_i, h(D_i), ID_S \rangle$ from his timeline, in order to enforce the verification protocol to check that the confirmation message $\langle y_i, h(D_i), ID_S \rangle$ is linked in S (see 2 of the signature verification protocol) and thus to respond `(invalid, unsafe)`. This way, the repudiation of the signature on D_i would succeed. However, the signature generation protocol enforces the master to verify that the message $\langle y_i, h(D_i), ID_S \rangle$ is linked in S, before confirming it (see 1 of the signature generation protocol). Thus, thanks to Assumption **A7** (ensuring that a fake chain source cannot be simulated by the signatory), this attack is not possible.

(3) The third mode leverages the latency of the confirmation tweet. In detail, let $\langle y, h(D), ID_S \rangle$ the tweet corresponding to the signature the attacker tries to repudiate and suppose that this message is linked in S. To do this, immediately after the previous tweet the attacker publishes a fake tweet $\langle y, h(D'), ID_S \rangle$. When he want to repudiate the signature on D, he deletes the message $\langle y, h(D), ID_S \rangle$. Observe that the attack succeeds only if the signature on D appears valid until it is repudiated (thus apparently satisfying the related obligation). The aim of the attacker is to jeopardize the recipient-side check based on timestamps described in item 2(a)ii of the signature verification protocol, by trying to obtain that the timestamp of the confirmation tweet associated with $\langle y, h(D), ID_S \rangle$ is more recent than the timestamp of $\langle y, h(D'), ID_S \rangle$. Indeed, in this case, the check described in 2(a)ii of the signature verification protocol would output *(valid, unsafe)* on D' (since the signature message is linked) and *(invalid, unsafe)* on D, thus allowing the repudiation of the signature on D. However, the above situation cannot hold. Indeed, three cases may occur. The first case is that the attacker is able to publish the second message $\langle y, h(D'), ID_S \rangle$ within Δt from the timestamp of the message $\langle y, h(D), ID_S \rangle$ (recall that Δt is the reaction time constraint defined in Section 6.2). In this case, according to option 2 of the signature generation protocol, the master publishes an aborting messages for y. Therefore, according to item 2(a)i, the signature verification protocol returns *(invalid, unsafe)* for both documents (since they are both inside the interval Δt), thus inhibiting the signature on D and thus vanishing the plan of the attacker. The second case is that the attacker publishes the second message $\langle y, h(D'), ID_S \rangle$ after Δt from the timestamp of the message $\langle y, h(D), ID_S \rangle$. In this case, according to item 1 of the signature generation protocol, a confirmation message $\langle y, h(D), ID_S \rangle$ has been published by M before $\langle y, h(D'), ID_S \rangle$, due to the reaction time constraint. Consequently, the signature verification protocol will return *(valid, unsafe)*, as explained in item 2(a)ii. The repudiation cannot be done. The third case is that the schedule of the posting-reaction sequence is as follows. First the attacker publishes the message $\langle y, h(D), ID_S \rangle$. Within Δt, the master generates the confirmation for this message and, in the meanwhile, the attacker publishes the second message $\langle y, h(D'), ID_S \rangle$. This would trigger the publication by the master of an aborting message for y thus falling into the first case above.

Resistance to Attack AD3. Recall that this attack occurs whenever an adversary tries to repudiate a social signature by jeopardizing the revocation system. Specifically, the adversary, after signing a document, performs a revocation request to claim that the revocation occurred before the signature. As both signature, confirmation and revocation messages include a trusted issuing time, the security against this attack is guaranteed by Assumption **A4**.

Resistance to Attack AD4. Recall that this attack occurs whenever an external adversary (different from the signatory) tries to make a valid signature generated by a certain user be verified as invalid. To do this, the attacker should remove the signature message from the victim's social network page and the confirmation message from the master social network page. According to Assumptions **A4**, **A5**, and **A1**, our system is secure against this attack.

Resistance to Attack AD5. Recall that this attack occurs whenever an adversary tries to make a signature generated over a fraudulently modified document be verified as valid. To do this, the attacker should inject a signature message into the victim's social network page or the corresponding confirmation message into the master social network page. According to Assumptions **A4**, **A5**, and **A1**, our system is secure against this attack.

Resistance to Attack AD6. Recall that this attack occurs whenever the master, playing as adversary, tries to make invalid a valid signature verification by illegal operations (w.r.t. the signature generation protocol) on his timeline.

This attack can be performed in three modes:

(1) The first mode is tried by the master by deleting the confirmation message associated with a linked signature message $\langle y, h(D), ID_S \rangle$. This attack fails because the signature verification protocol returns the output `(valid, unsafe)`, according to item 2(a)i.

(2) The second mode is tried by the master by deleting all the confirmation messages successive to the victim linked signature message $\langle y, h(D), ID_S \rangle$ and, then, by posting an aborting message for y. This attack fails because the reaction time constraint will be not satisfied for the aborting message, as described in the case 2(a)ii of the signature verification protocol. Indeed, the output returned by the verification protocol is `(valid, unsafe)`.

(3) The third mode to be considered is a fake revocation message. This attack cannot succeed thanks to Assumption **A8**.

Now, we conclude by considering security against attacks on document immutability, that is security property SP4 of the security model SM.

Theorem 6.4 (Document Immutability). *The protocol is secure against attacks AP1, AP2, and AP3.*

Proof. Recall that attack AP1 occurs whenever an adversary includes dynamic contents in a document being signed, attack AP2 whenever an adversary corrupts the fonts of the host system, and attack AP3 whenever an adversary produces a polymorphic file [36, 151] as document being signed. As widely proven by the literature [118], all CADES/XAdES qualified electronic signature systems compliant to [1] are not resistant to attacks AP1 and AP2. This is true also for attack AP3. In this case, the inclusion of the content-type of the document among the authenticated attributes of the CADES/XAdES format would solve the problem [118]. Similarly, in our scheme, we just would compute the digest of the document being signed by applying the cryptographic hash function to the concatenation of the binary content of the document with its content-type. Thus, according to Assumption **A1**, we conclude that our system is secure against attacks AP1, AP2 and AP3.

6.4 A Twitter-Based Instantiation of our Model

In this section, we describe the instantiation of the model presented in the previous section in a real-life scenario. The aim of this section is to prove that the idea embedded into the abstract model, besides being an original and new approach, has a practical value, as it leads to a concrete solution whose benefits we expect could be relevant both in terms of user experience and economic aspects. In our instantiation, the actors of the model are the following. The role of signature master is played by any private or public organization of any size, which we call generically as `Company`, which applies the electronic signature solution to all documents exchanged with its internal members, which we denoted as `employees` (and among them), who play the role of signatories. The role of social network is played by `Twitter`. The reason underlying the choice of `Twitter` derives both from its popularity and from the fact that it is centered on `tweets`, which are the only way to share public content. Our choice allows us to map a signature message to a tweet. The instantiation of our model in the `Twitter` setting leads to a concrete electronic signature solution that we have called *Tweet to Sign* (T2S, for short). In the remainder of this section, first we describe the specificities of `Twitter` used by our system, as they strongly influence the design choices. Then, we present our solution.

6.4.1 `Twitter` Specifics

`Twitter` is a microblogging services made up of 140-character messages called *tweets*. It is an easy way to discover the latest news coming from other people. It was designed to fit into the character limit of a text message, and `Twitter` still works on any SMS-ready phone. Brevity keeps `Twitter` fast-paced and relevant by encouraging people to tweet in the moment and to focus on the essential idea they are trying to communicate. Inside a tweet the user can see photos and videos from people he knows or behind-the-scenes moments from the biggest stars. The user can link to news stories, blogs, websites and apps. Once the user tweet is generated, it is publicly posted on his `Twitter` profile. The stream of tweets of a user is called *timeline*. A user can *follow* another user and becomes a *follower*. Tweets of a user appear in the timeline of their followers and are called *retweets*. Each `Twitter` user is identified by a username starting with the symbol `@`. People use `@` to mention a person in tweets. To categorize tweets by keyword, people use the hashtag symbol # before a relevant keyword or phrase (no spaces) in their tweet. Moreover, hashtags are indexed to make it easier to find a conversation about that topic. Applications and websites which use `Twitter` are built using the `Twitter` API. There are three kinds of API: (i) The REST API is used to do things like post tweets, follow someone, create lists and more; (ii) the Search API is for performing searches; (iii) the Streaming API is for application developers who want to receive a real-time stream of the public tweets on `Twitter`. The official guide to `Twitter` [2] provides more detail on these aspects.

6.4.2 T2S: Tweet to Sign

In this section, we describe our Twitter-based solution instantiating the model presented in Section 6.2.

T2S Actors. As introduced earlier, the actors of our system are the Company, which is the entity that needs to adopt the electronic signature (signature master), and its employees, who are the users signing documents (signatories). As required by our model, it is possible to extend the domain with other employees at any time. The role of the social network T is played by Twitter. It is easy to recognize that Assumptions **A3** and **A4** are realistic for Twitter. Concerning the estimation of the reaction time constraint Δt, we refer the reader to Section 6.6.

T2S Posting integrity mechanism. It is implemented by means of a hash chain computed by SHA-256, here denoted by h. Specifically, each employee S generates a secret p from which the element y_0 is computed as $h^u(p)$, where u is a suitable large value defined in the security parameters (see below), and h^u denotes the application of the self composition of the function h to p. This way, given y_i, it holds that the computation of y_{i-1} is efficiently done by computing $h(y_i)$. On the contrary, given y_{i-1} the computation of y_i is feasible only for S by computing $h^{u-i}(p)$. Thus, we have built a trap-door function with secret p that, thanks to the properties of SHA-256, is also second-preimage resistant. The secret p is managed in accordance with Assumption **A9**. We will see the technical detail in Section 6.5.

T2S Registration Protocol. As required by the model a Twitter account for the Company is created by a person who is authorized to act on behalf of the Company. Observe that Twitter allows for double factor strong authentication [8]. Assumption **A5** is thus realistic. Assume that the username chosen for this account is @Company. Next, each employee signs up to Twitter and, suppose, he uses @emp as screen-name. In this phase, the Company is responsible of the verification of the employee identity, which, due to Assumption **A6**, must be done in-person. Then, on Twitter he adds a follow relationship to @Company and, similarly, @Company follows back the employee account. The employee registration message is the tweet: $\langle$ #y_0 I am an employee of @Company $\rangle$, where y_0 is is obtained by the posting integrity mechanism above. The Company welcome message is $\langle$ #y_0 #h(emp) @emp is Y $\rangle$, where y_0 is the same as the registration message, h(emp) (which is hashtagged) is the cryptographic hash (SHA-256) of the screen-name emp of the registered employee and Y is an information identifying the employee himself. Y is typically the name and surname of the employee; however, further information, such as the employee id, is added to manage cases of homonymy. In this phase, all documents and actions needed to guarantee Assumptions **A6**, **A7**, and **A8** are established.

T2S Revocation Protocol. The revocation tweet posted by Company is $\langle$ @emp is revoked for I $\rangle$, where I represents the information regarding the reason of the revocation. Note that the time of the revocation coincides with the time of the tweet.

T2S Signature Generation Protocol. Suppose that @emp is the signatory and D be the document being signed. The hash function h used to compute the digest of the document is still SHA-256. This ensures that Assumption **A2** is realistic. The tweet representing the linked signature message on D has the form: $\langle$ #y_i I signed the document #h(D) $\rangle$, where y_i is

$h^{u-i}(p)$ (i.e., is the $(u-i)$-th element of the posting integrity chain). The confirmation tweet posted by the Company is: ⟨ `#`y_i `@emp signed the document #h(D)` ⟩. Observe that the ID of the signatory @emp (corresponding to ID_S of the model) is not reported in the signature messages because it is implicit. This reduces also the set of attacks considered in the security model described in Section 6.3. In particular, the third mode of the attack AA5 described in Section 6.3.2 is not feasible. The aborting message posted by the Company in the case 2 of the signature generation protocol has the form: ⟨ `#`y_i `of @emp is aborted` ⟩.

T2S Signature Verification Protocol. Twitter allows us to implement the signature verification protocol in a very efficient and natural way. Basically, the operations required by the protocol are: (1) the searching of a signature tweet ⟨ `#`y_i `I signed the document #h(D)` ⟩ or a confirmation tweet ⟨ `#`y_i `@emp signed the document #h(D)` ⟩, on the basis of $h(D)$ to find the occurrence of a signature on D and on y_i to check that the message is linked, (2) the searching of an aborting tweet (in the case 2(a)i of the signature verification protocol) ⟨ `#`y_i `of @emp is aborted` ⟩ on the basis of y_i, (3) the searching of a registration tweet ⟨ `#`y_0 `I am an employee of Company` ⟩, on the basis of y_0, in the worst case of the check of the linkness of a signature message, (4) the searching of a welcome tweet ⟨ `#`y_0 `#h(emp) @emp is Y` ⟩, on `h(emp)` to resolve the real-life identity of the signatory, and (5) the verification of the reaction time constraint compliance required in the case 2(a)ii of the signature verification protocol.

Observe that all the above operations but the last one are done on information which is hashtagged in our tweets, so it is feasible and efficient. The last operation is done by checking the timestamps of the tweets.

T2S Security Parameters. The security parameters $LoA(Auth)$ and $\langle LoA(Id), p\rangle$ are set to 3 and $\langle 3,1\rangle$, respectively, meaning that only two factor authentication and in-person identification in the registration phase in which possession of verified current primary government picture ID that contains also applicantÂŠs picture are allowed [66]. Our solution includes another parameter u, which is a number representing the maximum number of signatures a user may generate (as discussed in the definition of the signature generation protocol in Section 6.3, u can be large enough to cover all possible signatures of a user). As we have seen earlier, this parameter is related to the posting integrity mechanism implemented as an hash chain of length u.

6.5 Implementation of the proposal

In this section, we describe in detail the technical aspects concerning the architecture and the implementation of our solution. The T2S application is composed of five independent software packages: (1) two user-side packages, allowing the hash chain generation and the document digest verification, resp.; (2) the signature software, which retrieves from the signatory all the information necessary for a new signature and interacts with Twitter to publish the tweet associated with the signature; (3) the verification software, which returns the list of employees who signed a given document; (4) a Company-side software service, which implements the automatic generation of the confirmation tweet. Concerning the signature and verification software packages, we integrated them into the same application that has been built both as stand-alone application and as Web application. The Web application is the most complete. Let us proceed by describing in detail the above components.

The user-side software packages. T2S provides two user-side software packages, namely an application for the generation of the hash chain and a Firefox plug-in to validate the document digest. As stated in Section 6.4.2, in accordance to the requirement of the posting integrity mechanism, the former software must be resistant to attacks based on key-loggers (Assumption **A9**). For this reason, we chose to implement it as a mobile phone application and, in particular, as an Android app. The compliance with Assumption **A9** is obtained by the fact that, as stated in the Android specifications [6], the execution of an Android application follows the *principle of least privilege* [84] for which each application runs in a separated Linux process inside an independent virtual machine. This mechanism prevents the leakage of data during the lifecycle of an Android application. Moreover, the permanent storage of data can be kept private inside a memory space reserved to a single application, so that no interaction with external software is allowed [7]. Observe that this *sandbox* security mechanism could be in principle compromised *only* by unlocking root privileges on the smartphone. This non-trivial operation may allow an attacker to by-pass the Android OS layer and access the underlying Linux environment. However, unlocking root privileges on a smartphone is not a legitimate procedure. Indeed, for this reason, almost all vendors invalidate their product warranty if such an action is carried out on the device. Therefore, we assume that a signatory cannot use an unlocked smartphone to run our software. The Android application keeps the user secret (inserted during a secure initial step by the user himself) along with the number i of generated elements in its private memory. Each time the user signs a document, the application generates the entire hash chain and returns the $i+1$-th element. As for the Firefox plug-in, it verifies if the generated document digest is correct. This plug-in computes the SHA-256 of the document sent by the user to the signature software and compares it with the content of the tweet generated by the signature application. This is a further security functionality provided by our solution.

The signature and verification software. This software is implemented as a Java Web Application and runs on a server equipped with Apache Tomcat Server. It is composed of three main modules: (1) the `Authenticate` (sub-)module, which handles users authentication; (2) the `Sign` (sub-)module, implementing all the tasks necessary to sign a new document; (3) the `Verify` (sub-)module, in charge of performing the steps required to identify the signatories of a given document. The module `Authenticate` is activated once the user inserts his credentials to access the Web application. After the authentication, the user chooses the action to do: either signing a document or verifying a signed document. The module `Sign` relies on three components, which are shared with the other modules of the Web application: (i) the `Digest` component, which generates the digest of the input document, (ii) the `TweetGen` component, whose purpose is to convert the digest generated before into a string and to prepare the body of a new tweet, and (iii) the `Share` component, in charge of posting the new tweet on `Twitter`. The module `Sign` is activated whenever the user chooses to sign a document. After the user selects the file being signed, the SHA-256 cryptographic hash function of this file is computed. The generated digest is then encoded into a base64 string that will be inserted in the tweet. Subsequently, the user is asked to insert the element y_i of his hash chain. We remaind that the user can generate it and automatically send it by means of the Android application described in Section 6.5. Then, a new signature tweet is generated and validated by the Firefox plug-in (see Section 6.5) and posted on `Twitter`. In particular, the posting action is done by calling a specific link which is normally used to implement the `Twitter` "share button". For instance, suppose that:

$$h(D) = \texttt{FtC7d6XTSJOK5HCmQwzFeLtnkDDbmA/vOOhviZ58eQQ}$$
$$y_i = \texttt{9uJDZH7mGB2Fn4RqXOB9jrpNP3q3z4EsYPklbm8+WMc}$$

then this link has the following structure:

```
https://twitter.com/intent/tweet?button_hashtag
= 9uJDZH7mGB2Fn4RqXOB9jrpNP3q3z4EsYPklbm8+WMcF
tC7d6XTSJOK5HCmQwzFeLtnkDDbmA/vOOhviZ58eQQ&text
= I have signed the document
```

where the request parameter `button_hashtag` contains both y_i and the document digest. The parameter `text` is the rest of the tweet (see Section 6.4.2). The verification of a document digest is performed by means of the module `Verify`. This module makes use of the following components: (i) the `Digest` component described earlier, (ii) the `Search` component which executes a `Twitter` search of an hashtag and handles the results to extract information on the users who posted tweets including this hashtag, (iii) the `CheckTweet` component, in charge of verifying whether the tweet represents a valid signature as described in Section 6.2, (iv) the `RevealEmp` component, which solves the real name of each tweeter identified as signatory. This module receives the document whose signature has to be verified. Then, the digest of this document is computed by using the same cryptographic hash function as the signature generation (this task is performed by the `Digest` component). Hence, the digest is used as hashtag to perform a `Twitter` search to find all the users who have generated a tweet including this hashtag. Observe that this task cannot be performed by relying on `Twitter´s` API. Indeed, as specified in the `Twitter´s` API documentation [9], search APIs do not return all tweets but only the last tweets of a week. According to `Twitter`, the collection of all tweets referring to an hashtag should be performed by using the Streaming APIs. This way, it is possible to receive all the tweets posted with the observed hashtag in "real-time". However, this solution does not suit our application context: indeed, by relying on the Streaming APIs a trusted party should store all the tweets generated by `employees` to build a database to be used for the signature verification. This is not applicable to our case in which no trusted third party (besides the social network itself) is required. Thus, in our solution, the search is performed through the `Twitter` Web interface, in which no limitation on the number of retrieved tweets exists. For this reason, the `Search` component is based on an HTML parser analyzing the results of queries to an ad-hoc modified link. For example, considering again the document digest above, the search link becomes:

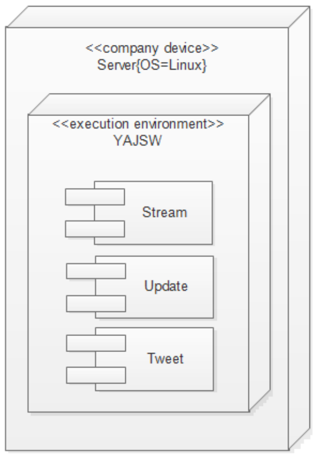


Fig. 6.3: The deployment diagram of the T2S Service.

```
https : twitter.comsearch?f = realtime&q = %23FtC
7d6XTSJOK5HCmQwzFeLtnkDDbmAvOOhviZ58eQQ&src =
typd.
```

The parsing of the results of these queries allows the extraction of the IDs of the tweets having the document digest as hashtag, the screen-names of the users who posted them, and the screen-name of the users mentioned in the tweet. After this, the Verify module distinguishes tweets generated by the Company from those generated by employees. This way, by following the procedure described in Section 6.2, this module identifies valid signatories. Finally, the real name of signatories has to be extracted. This is done by leveraging again the Search component to find the Company tweet linking the screen-name of an employee's Twitter-account to his real name, by using the hash of the screen-name as hashtag (see Section 6.2).

The T2S Company **Service.** The aim of this software is to implement the tweet mechanism that allows the Company to "confirm/abort" the signature of documents done by its employees. It is implemented as a Java application and runs as a Unix daemon under the YAJSW wrapper tool. This application consists of three modules: (1) the Stream module, which receives new tweets from the Company followings (employees), (2) the Update module, which allows users to send an update request whenever some changes are made on the Twitter account of the Company, and (3) the Tweet module whose aim is the automatic re-share of each tweet posted by an employee.

Since the application is installed as system service, the modules Stream and Update are activated on system boot. In particular, once executed, the Stream module opens a new connection and associates a listener with the Twitter Streaming APIs. According to the default configuration, this listener receives a random sample of the tweets generated globally on the platform. However, it is possible to add a filter to the Twitter streaming connection so that the listener will receive only tweets coming from a specified set of accounts. In our case, we built a filter allowing the listener to receive information coming from all the accounts in the list of the Company followings. The list of followings is automatically retrieved through the

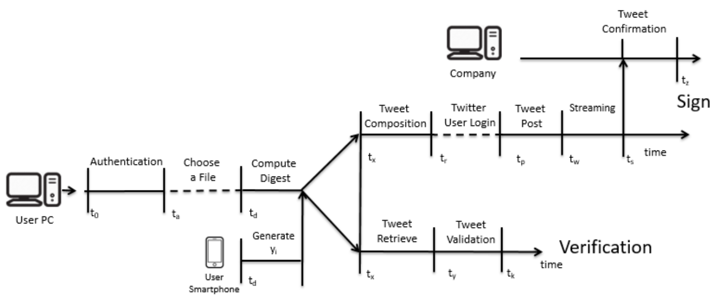


Fig. 6.4: Generic timeline of T2S actions.

Twitter Rest API GET friends/ids. However, if some modification happens on this list (e.g., a new employee is added) while the T2S Company Service is running, a new query to this API method will be submitted by means of a trigger implemented in the Update module. Once a new tweet posted by one of the employees is filtered out, the module Stream uses the functionalities provided by the module Tweet to share a new tweet built according to the policies described in 6.2. To do this, this module uses the Twitter Rest API POST statuses/update.

6.6 Performance Evaluation

In this section, we provide an experimental campaign to test the performances of our system in terms of time overhead. The experiments had been performed on a personal computer equipped with 4.0GHz Intel i7 CPU, and running Ubuntu 14.04 with Apache and Tomcat servers. As for the client, we used a personal computer with the same hardware features. All the tests had been carried out with the browser Mozilla Firefox version 14.0.1 and, besides Firefox, no other user applications were running. The overall time diagram is shown in Figure 6.4. It deals with the two main functions of our prototype, which are: Sign and Verification.

These two actions have some preliminary steps in common. The former is the Authentication, in which the user sign in the system. We tested the duration of this first step and we found out that it lasts less than 50 milliseconds on average. After this security step, a user is prompted to choose the file to sign or verify from his PC. This step has to be carried out manually, so we do not consider it in our analysis. Then, the algorithm to get the digest of the document is executed. Clearly, the overhead of this step depends on the size of the file chosen. Hence, we run a set of experiments to investigate how the algorithm execution time varies with the size of the file chosen. We measured the time overhead for files in a range of size from few kilobytes to 250 megabytes and obtained an average execution time of 4.08 sec.

The actions discussed above totalize a time overhead equal to $t_x - t_0 \simeq 4.13$ seconds, where t_0 is the initial time. From now on, we consider different actions and, consequently, different time overheads, depending on whether the user signs or verifies a document. Consider the case in which a user signs a document. In this case the software has now to generate the tweet to be shared on Twitter automatically. This procedure is composed of two sub-steps. The former is the generation of the element y_i of the user hash chain. This is done by means of the

Actions	Time (milliseconds)
Authentication $(t_a - t_0)$	41
Signature-tweet posting $(t_w - t_d)$	3300
Company *Reaction Time* $(t_z - t_w)$	3600
Signature verification (best case) $(t_k - t_d)$	3600
Signature verification (worst case) $(t_k - t_d)$	7200

Table 6.1: Time overhead of T2S actions.

Android application on the user device. We measured that the y_i generation time is negligible whereas the device takes about 1.8 seconds to send this value to the Web application hosting the signature software. The latter is the tweet generation. We obtained that the time for this step is almost 0. After this, the user has to sign in Twitter. This step has to be carried out manually so, once again, we do not consider it in our analysis. Finally, we have to consider the tweet posting execution time. We measure that this time is about 1.5 seconds. Subsequently, we measured the *streaming* time, i.e., the time interval required by the Twitter Streaming API to propagate this tweet to the Company software for the generation of the confirmation/aborting message. We obtained that the streaming time is 2.1 on average, whereas the procedure for the generation (including all verifications required by the protocol) and posting of the Company tweet takes 2.5 seconds on average. This time interval, say $\Delta t = t_z - t_w$ (see Figure 6.4), represents a good estimation of the reaction time constraint defined in Section 6.2. On our testbed, it is about 3.6 seconds. As for the Verification step, the time overhead to check the signature of a file is $t_y - t_x$, in which t_y is the time required to find a tweet containing a given hashtag, to verify if this tweet is a valid signature message, and to retrieve his real name (see Section 6.2). We measured that the search for a given hashtag and the extraction of signature tweets from the result lasts about 1.8 seconds. Whereas, the time for the validation of a single signature tweet ranges from 1.8 to about 5.4 seconds on average. This variation of the estimated times is due to the additional checks required by our model in case of anomalies in the validation of a signature (see Section 6.2). The results of our performance measurements are summarized in Table 6.1.

6.7 Discussion

The concept of electronic signature is more and more important in a society moving towards a complete dematerialization of documents and transactions, both in the public sector and in business. The protocol we have presented is thought by keeping in mind the above dynamics. Our work is an attempt to give an innovative, practical, and secure solution of advanced electronic signature overcoming the drawbacks of cost and low usability of qualified electronic signature. We obtained this by relying on the power of social networks of sharing information among people. The starting question was: Can we exploit what social networks offer *for free* to play strategic functions of organizations and communities in a secure, cheap, and reliable way? So far, social networks have been mostly used as communication media. We think they can become part of people workflows, with reciprocal advantage: for people high usability, low cost, high availability, for social network providers more central role in the society. In this chapter, we tried to apply this new paradigm to electronic signature. We addressed the case in which an entity exists able to identify and register signatories at the initial stage, with no limit in terms of its dimension. The scalability of our solutions arises from two facts. The first one is that social networks, by definition, provide scalable services. The second one is that our protocol, differently from the current qualified electronic signature, does not make use of public-key encryption, but only cryptographic hash functions (which are inherently efficient) and other simple operations. The overhead computation for the signature master is thus feasible. In other words, the pure adoption of social networks in a e-signature framework using public-key encryption would be much less realistic than our solution. Another nice feature of our social-network-based signature is that even though signatories must be previously identified and registered, recipients can be everywhere and anyone in the world, thanks to the pervasivity of social networks. Indeed, the verification of signatures leverages only public information available on a social network and easily searchable. The solution is also privacy-preserving, in the sense that the universal exposure of signatures does not affect the privacy of signed documents, as only document digests are published. Concerning the aspect of privacy, it is worth noting that we consider the case in which it is not a privacy threat that an adversary may discover that a given document has been signed by a certain user. Therefore, this attack is not contrasted. However, the current version of our protocol is thought for truly sharing environments, where only the content of documents should be protected (also because the published information is accessible by everyone, even external to the sharing community). However, if the above conditions do not occur, we just have to modify our signature protocol by including some salting-based techniques in the computation of digests. In this case, the price we have to pay is that the set of recipients of a signed document must be prefixed (only those subjects equipped with the information needed to detect the salts). Observe that this is not a limitation. It is an intrinsic feature, because any recipient which is able to verify a signed document can perform the attack above, independently of the technique used for signatures. Also timing attacks (based on deadlines, periodic signatures, etc.) to discover if a (even salted) document has been signed by a certain user can be contrasted by generating dummy traffic in the timelines of users. The implementation of this extension is subject of our future work. Our protocol provides users with another good feature. Indeed, our protocol supports implicitly a signature timestamp service which is secure and enforceable against third parties, with no extra cost. Recall that, to do this, the current qualified electronic signature requires the adoption of additional paid services, typically provided by certification service providers, also reducing the usability of signature applications (timestamps are generated by additional operations). Another strong point of our signature is that multiple signatures on a single document are implemented in a very easy and flexible way, with no need of planned exchanges of the document being signed, as it happens for existing qualified electronic signatures. The last nice feature we want to highlight is that our solution is proven to be at least secure as

CADES/XADES signatures (which are the state-of-the-art qualified electronic signatures), in a security model in which only realistic assumptions are done and everyone (but the social network provider) can be the adversary (including the signature master). The complexity of the security analysis (together with the high granularity of our signature verification procedure) shows also that the implementation of the apparently trivial starting idea of publishing digests of signed documents opens a number of security issues, which can be addressed, leading to a definitely non-trivial solution.

6.8 Related Work

In this section, we give a quick survey on the proposals related to electronic signature present in the literature, besides the most known public-key-encryption based signature relying on a PKI. Conditional Signatures were originally introduced by Lee et al. [152] to implement fair exchange of digital signatures in untrusted environments and do not require the card to have a user interface or any special peripheral (like Clarke et al. [78]). Berta et al. [31] propose a method to generate, instead of an ordinary signature, a conditional signature such that it is guaranteed that the condition can not become true before a certain amount of time has passed. This should leave time for the user to move to a trusted terminal for checking the signatures generated by the card and to enforce that the conditions of the fake signatures can ever become true. Weak signature was introduced by T. Rabin and Ben-Or [197] to solve a problem motivated by a question of general multi-party secure computation in the unconditional setting (network of untappable channels). They provide a form of authentication for which the on-line participation of a third party is needed. Another signature scheme in the unconditional setting was introduced by Chaum and Roijakkers [71]. It satisfies a stronger set of conditions than Rabin's Information Checking Protocol, at a great increase in communication cost. Naor et al. [178] suggest a number of transparency-based methods for visual authentication and identification, and give rigorous analysis of their security. [170] presents human-friendly identification schemes such that a human prover knowing a secret key in his brain is asked a visual question by a verifier, which then checks if an answer sent from the prover matches the question with respect to the key. Ateniese et al. [27] propose a visual cryptography scheme for a set of participants to encode a secret image into many secondary images in such a way that any participant receives one secondary image and only qualified subsets of participants can "visually" recover the secret image, but non-qualified sets of participants have no information, in an information theoretical sense, on the original image. This scheme does not require the participants to perform any cryptographic computation. In [212], the authors propose a inclusive security development that provides security for communication in the MODBUS protocol and trends for future development and improvement of SCADA systems. They selects and testes the most important cryptography algorithms to conclude that the platform provides remarkable performance. The model of signature presented appears innovative w.r.t. the state of the art. Indeed, as better explained in in the next section, it presents some peculiar characteristics which can be summarized as follows: (1) it is oriented to sharing environments, (2) it minimizes the implementation of dedicated infrastructures because it exploits existing social networks, (3) it is scalable and highly usable, (4) it does not requires the usage of encryption, (5) it does not rely on trusted third parties (besides the social network) and certification service providers, (5) it provides a timestamp service *for-free*, and (6) it naturally supports multiple signatures. Therein, the protocol is much simpler and not resistant to master-side attacks. Moreover, no theoretical model, deep security analysis, and implementation are provided.

7

Anonymous Authentication for Delivery of Cloud Services

Always in the context of e-government, a critical aspect is the balance between privacy of users when they access cloud services and accountability. The solution of this trade-off becomes essential especially in scenarios where user's privacy is threatened by honest-but-curious *cloud providers. Indeed, cloud computing is an emerging paradigm whose importance in e-government is more and more increasing. In this scenario, we propose an authentication scheme supporting full anonymity of users and unlinkability of service requests. This is done by combining a multi-party cryptographic protocol with the use of a cooperative P2P-based approach to access services in the cloud. As this solution is thought to be adopted in e-government scenarios, accountability of user accesses is always preserved to prevent misuse and illegal actions.*

7.1 Overview of the Proposal

In this section, we sketch out the idea underlying our proposal through a motivating example. Consider the case in which a user exploits an e-health services of a cloud provider to interact with an health-care institute of a given country and, then, in the same cloud session, makes a flight reservation for that country. This example is illustrated in Figure 7.1.

These cross-domain data can be combined by the cloud service provider, assumed honest but curios, to derive information on the private life of the user, and therefore, to obtain data the user was not meant to reveal. In this case, for instance, it is possible to infer that either the user is an employee of the health-care provider who is reaching his working place or a patient who is requiring hospitalization. Therefore, by analyzing the typology of health-care provider (e.g., mental hospital, orthopedic center, etc.), it is possible to make assumptions on user's disease.

Consider that, the sole application of anonymous authentication schemes like [169] is not sufficient to solve our problem. Indeed, the service provider may still obtain user data from the flight reservation and link them with the information on the health-care provider.

In our example, our technique proceeds as follows:

1. The user U sends his identity together with the identity of the cloud provider P to a grantor G, which typically is an e-government institution where the user is registered.

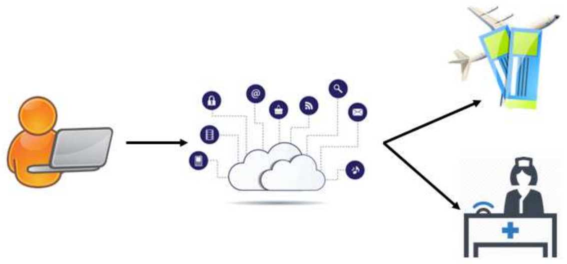


Fig. 7.1: A user accessing the cloud to have information about an hospital and to make a flight reservation.

2. G responds by sending some *tokens* and the reference to an entry point for the P2P network. Each token includes a *ticket* (i.e., a credential) and a key. The ticket is spent for the service, the key is maintained secret.
3. The user joins the P2P network and uses this network to send two tickets (one for each request) to P anonymously.
4. P receives the two tokens from two users (different from U) of the P2P network, so that the requests appear anonymous and unlinkable. Each ticket contains information that only G and P can decrypt to establish that the credential is valid and to extract a secret key for the secure communication with the user. The service is thus ciphered by this key and delivered to the user by using again the P2P network.

In summary, our approach leverages on three basic features: anonymous authentication, unlinkability of user service requests, and traffic flow anonymity in the communication with the cloud service provider. Specifically, the first one is achieved by relying on a solution like [72, 169] which leverages on the interaction with a grantor, playing the role of trusted third party, to perform anonymous authentication to cloud services. Concerning the unlinkability of service requests, our solution works by assigning different tokens for each request, thus decoupling cross-domain information. However, since the cloud provider may still associate service requests with the IP address of the user, we adopt a strategy leveraging on a P2P network for the IP obfuscation. It is worth noting that only the combination of the two strategies (i.e., multiple anonymous tokens and P2P user interface) achieve the privacy goal.

Finally, we discuss about accountability. The provider P logs all user's activities by associating them with the random number included in the corresponding credential. Thus, no information can be drawn from the analysis of logs even about behavioral patterns of the user. Only in case of need (for example, in case of illegal actions), logs can be linked to the identity of the user by using information kept by G, thus allowing full accountability. In the next section we describe in detail how the protocol is defined.

Symbol	Description
U	A user accessing a service
P	The cloud service provider
G	The grantor
$\mathcal{N}_{\mathcal{S}}$	Nodes of the P2P network
A	An entry point of the P2P network
$\mathcal{H}$	Cryptographic hash function
$E^Q(x)$	The encryption of x with the key of an entity Q
$D^Q(x)$	The decryption of x with the key of an entity Q
$\|\|$	Concatenation operator

Table 7.1: Notations.

7.2 The service delivery protocol

In this section, we describe the design of our protocol for anonymizing the access to cloud services. Preliminarily, we report in Table 11.1 the notation used throughout the rest of the chapter.

The protocol we propose relies on an underlying P2P network, which is used to anonymize communications. In particular, this avoids that the cloud service provider may obtain useful information from the analysis of IP addresses of users accessing the services.

The entities involved in our protocol are:

1. The user U who needs to access cloud services.
2. A trusted third party said *grantor* G, which identifies users and provide them with *tickets* necessary to enjoy cloud services.
3. The *provider* P, which supplies cloud services.
4. The open ended set $\mathcal{N}_{\mathcal{S}}$ of the nodes of the P2P network.

According to our protocol, a public key infrastructure exists so that both grantor and providers have a certificate containing a public key. The protocol is structured as follows.

Initial Registration. The user U is identified and registered by G. All necessary information to establish a secure channel is now exchanged.

Identification. In this phase, the user U submits his identity to the grantor G via secure channel established in the initial registration. Moreover, U sends G the public key certificate of the provider supplying the services he wants to access.

G verifies the identity of the user and his authorizations and grants a set $\mathcal{TK}_{\mathcal{S}}$ of n pairs (where n is suitably set system parameter) $(ticket, key)$ and the reference to a node $A \in \mathcal{N}_{\mathcal{S}}$. In particular, $\mathcal{TK}_{\mathcal{S}} = \{(T_i, K_i) : T_i = E^P(\tau_i \| r_i) \wedge K_i = \mathcal{H}(\tau_i \| r_i)\}$, where τ_i is a (long) validity time, r_i is a nonce, $E^P(x)$ denotes the encryption of x with the public key of the provider P (obtained from the certificate of P), and $\mathcal{H}$ is a cryptographic hash function. Moreover, each ticket T_i is signed by G to guarantee authenticity and integrity of the ticket. Observe that the value of n actually sets the overall number of requests the user can do without re-contacting G. As the size of each pair (T_i, K_i) is small, we can imagine to have a large n to drastically reduce the number of messages exchanged between U and G.

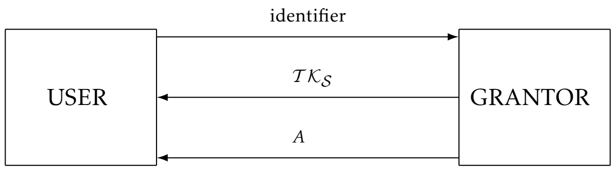


Fig. 7.2: The identification phase.

Concerning the node A, it is randomly selected from the last t users who have been authorized by the grantor to access some cloud services, where t is a system parameter set up according to the P2P network dynamics (this approach is aimed at maximizing the probability of finding the entry point alive).

Figure 7.2 summarizes the messages exchanged between user and grantor in this phase.

Service Request. Once the user has obtained the credentials to anonymously access the cloud, he can require a service to P. First, U joins the P2P network by using A as entry point. Then, U generates a secret S and computes $c_i = E^{f(K_i)}(S)$ encrypting the secret S with a key obtained as function f (for simplicity not specified here) of the i-th key K_i. Then, he computes $v_i = E^{f(K_i)}(t_i)$, where t_i is the timestamp recording the current time. Now, U creates the service request message m for P having the tuple $\langle T_i, c_i, v_i \rangle$ as authentication credential. This message is sent through the P2P network to reach P with an anonymous IP address. Concerning the use of a P2P network to obtain such an anonymity, we observe that there exist several approaches such as [245, 200]. It is worth noting that one of the most simple ways to obtain this goal is as follows. Each node of the P2P network receiving a service request message m for P, with a certain probability delivers the request to P, otherwise forwards the request to another P2P node. Involved nodes maintain the previous hop of the message route, which is used to delivery the reply coming from P.

The use of the P2P network allows for the creation of an anonymous tunnel which varies for different service requests coming from the same user. This way, the provider cannot link the tickets adopted by the same user to access different services from the knowledge of the sender IP address.

Once the provider receives the anonymous message originally generated by U from the P2P network, it verifies authenticity and integrity of the ticket using the public key of the grantor, and then deciphers T_i with its private key, thus obtaining τ_i' and r_i'. Therefore, it verifies the (long term) temporal validity of the ticket checking that τ_i' is less than the current time. Then, it verifies that the nonce r_i' has been never received in the past. If the ticket is expired or already used, the request is denied. Observe that the long term temporal validity is used in case of authorizations with validity time that must be reflected in the credentials sent by G to U.

At this point, P computes $K_i' = \mathcal{H}(\tau_i' \| r_i')$ and uses $f(K_i')$ as symmetric key to decipher c_i and v_i, thus obtaining the secret S and the timestamp t_i. If $t_i + \Delta t$ is less than the current time (where Δt is a general system parameter set to a small value for security reasons – see Section 7.3), the request is discarded. Otherwise, the protocol proceeds and if $K_i' = K_i$ (i.e., it is the correct key), then this key is valid. At this point, the provider uses this information to

Step	Messages
Identification	$U \rightarrow G$:identity
	$U \rightarrow G$:public key certificate of P
	$G \rightarrow U : \mathcal{TK}_{\mathcal{S}} = \{(T_i, K_i) : T_i = E^P(\tau_i \| r_i) \wedge K_i = \mathcal{H}(\tau_i \| r_i)\}$
	$G \rightarrow U : A \in \mathcal{N}_{\mathcal{S}}$
Service Request	$U \rightarrow A$: join $P2P$
	$U \rightarrow P2P : m = \langle T_i, c_i, v_i \rangle$
	$P2P \rightarrow P : m = \langle T_i, c_i, v_i \rangle$
	$P \rightarrow P2P$: encrypted communication
	$P2P \rightarrow U$: encrypted communication

Table 7.2: The messages exchanged in the protocol.

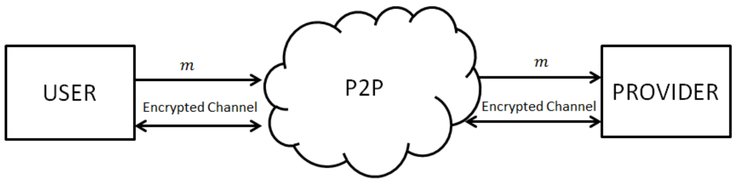


Fig. 7.3: The Service Request phase.

establish an encrypted communication (by using S) with the anonymous initial node through the P2P network.

Figure 7.3 summarizes the messages exchanged between user and provider in this phase. A schematization of the procedures executed by the actors involved in our protocol is reported in Algorithm 4. Table 7.2 summarizes all the messages exchanged in the protocol.

7.3 Security Analysis

In this section, we will show the robustness of our protocol against the most common types of attacks. This is discussed in the following.

Replay attacks. This type of attack is done by maliciously re-sending a ticket to access a service. If the ticket has been already spent from the legal owner, then it will be detected as not valid because the nonce r' has been already received. Another possibility is that the attacker intercepts the ticket when it is sent from the user to the provider. The expiration time Δt forces the attacker to use this ticket immediately because otherwise the ticket expires. However, the attacker cannot generate the correct secret S necessary to the communication with the provider because it is sent encrypted by $f(K)$. Moreover, as the messages exchanged between U and P are encrypted by S, the attacker has no advantage from intercepting and replaying them to any party.

Algorithm 4 Authentication and communication protocol

```
Procedure User-Side
 1: $U$ submits his identity to the grantor $G$
 2: $U$ sends the public key certificate of $P$ to $G$
 3: if (request is rejected) then
 4:     close
 5: end if
 6: $U$ generates $S$ and computes $c_i = E^{f(K_i)}(S)$ and $v_i = E^{f(K_i)}(t_i)$
 7: $U$ joins the P2P network by means of $A$
 8: $U$ sends $m = \langle T_i, c_i, v_i \rangle$ to $P$ via the P2P network
 9: $U$ establishes an encrypted communication channel with $P$
10: close
Procedure Grantor-Side
11: $G$ receives $U$'s request
12: if $U$ has invalid credentials then
13:     $G$ rejects the request from $U$
14:     close
15: end if
16: $G$ sends the set $TK_S = \{(T_i, K_i) : T_i = E^P(\tau_i \| r_i) \wedge K_i = \mathcal{H}(\tau_i \| r_i)\}$ to $U$
17: $G$ selects $A \in \mathcal{N}_{\mathcal{S}}$ and sends it to $U$
18: close
Procedure Provider-Side
19: $P$ computes $D^P(T_i)$ obtaining $\tau_i'$ and $r_i'$
20: if $\tau_i' <$ current time or $r_i'$ has been already received then
21:     $P$ returns false to $U$
22:     close
23: end if
24: $P$ computes $K_i' = \mathcal{H}(\tau_i' \| r_i')$
25: $P$ computes $D^{f(K_i')}(c_i) = S$
26: $P$ computes $D^{f(K_i')}(v_i) = t_i$
27: if $t_i + \Delta t \geq$ current time and $K_i' = K_i$ then
28:     $P$ establishes an anonymous encrypted communication channel with $U$
29: else
30:     $P$ returns false to $U$
31: end if
32: close
```

Spoofing attacks. The attacker simulates to be the grantor in order to obtain the login information of the user. The use of a PKI infrastructure for authentication of the grantor avoids this attack.

Password guessing attacks. In this case, the attacker tries to obtain the login information of the user by one of the following ways: (i) *on-line*, submitting possible authentication credentials until the grantor does accept the credential. As this attack needs the participation from the grantor, it is contrasted by including a delay in the reply of grantor to limit the number of attempts in the time from the attacker; (ii) *off-line*, in which the adversary guesses a secret without the participation of any other party. The secrets that he could guess are the followings. τ and r, because from them he can compute K. Although τ is easy to know as is a timestamp, r is randomly generated and as a consequence very hard to guess. Also the knowledge of K is hard, because it is a digest computed by a cryptographic hash function. Finally, the secret S is sent encrypted by $f(K)$ and then also in this case is very hard to guess. Clearly, in these considerations, we assume that the cryptographic functions and keys used are secure, as usual in this context.

Man in the middle. Here, the attacker monitors, alters or injects messages into the communication between the provider and the user who accesses the service. However, the secret S used to encrypt the communication channel between P and U cannot be known by the attacker. Moreover, he cannot alter the secret S sent to P in the *service request phase* because K (which ciphers s) is known by U and calculated by P. Thus, the attacker is not able to make them believe they are communicating directly to each other (condition necessary for the success of the attack).

Denial-of-service attack. This typology of attack is very wide and is carried out from the attacker by sending false requests to interrupt the service provided by P. However, as the service request messages are signed by the grantor, fake requests are easily detected. Moreover, in case a correctly signed message is sent to P more and more times, only the first requests will be accepted, and the others will be discarded. Thus, only one service will be provided with no possibility to overload the provider. Also the attack based on a prior man-in-the-middle attack by blocking and collecting a huge number of tickets and resend it as a burst to overload the service provider fails, because all tickets have a very small expiration time.

Behavior-based deanonymization attack. This type of attack is carried out by identifying recurrent patterns in the usage of services during an authenticated session. The knowledge derived from the analysis of service logs can be used to guess user identity on the basis of his attended behavior inside the cloud. This kind of attacks are contrasted by using different tickets for each service required. Moreover, because the attacker could associate requests coming from the same IP address, all messages to the cloud service provider are sent through a P2P network which adopts a routing protocol guaranteing anonymity.

7.4 Discussion

We have presented a new protocol for accessing cloud services in such a way that no information about users can be drawn from log analysis by a honest but curious cloud provider. The solution combines a multi-party protocol with a P2P approach to obtain anonymity at the granularity of the single user's request and unlinkability between different requests. Accountability is preserved, provided that the trusted third party cooperates. This assumption is coherent with the setting where this solution is thought, where the role of third trusted party can be naturally played by a government entity. Among the strengths of the proposal, besides its scalability and efficiency, we include the consideration that our solution has a realistic business model, as many e-government situations can be recognized where the public sector and the cloud provider market may have reciprocal advantages. The former has the advantage of outsourcing services towards the cloud, the latter has the possibility to exploit the (even attribute-based) digital identity management provided by e-government services. Concerning efficiency and scalability, at the stage of this research, we can argue that the solution appears good only on the basis of qualitative considerations (a few exchanged messages, simple cryptographic operations). Moreover, a possible extension of the P2P role can be done, by enabling multiplexing of the service delivery over multiple virtual (anonymous) channels. Both this extension and an accurate efficiency analysis are planned as the next step of this research.

7.5 Related work

Although a wide amount of work deals with general security issues in cloud computing, only few papers concern anonymous authentication. For instance, an overview of the different security risks that reduce the growth of cloud computing is presented in [215]. The authors of [237] focus on cloud data storage security issues, such as error localization and the identification of misbehaving server. The problem of ensuring the integrity of data storage is addressed in [238]. The authors consider the task of allowing a third party auditor to check the integrity of the dynamic data stored, on behalf of the cloud client. All the works cited above are inherently different from our approach because they do not focus on the definition of techniques for privacy-preserving access to cloud services.

A number of strategies ensure user privacy in cloud scenarios without relying on any cryptographic solution. In particular, [150] establishes a set of requirements for a secure and anonymous communication system and tries to fulfill those requirements by using a combination of existing systems, such as Tor and Freenet [79]. A client-based privacy manager that helps reducing the risk of data leakage and loss of privacy is proposed in [176]. However, the authors do not take into account the information derived from the possibility of linking different user sessions that may ultimately result in user profiling attacks.

Several works deal with data privacy concerns. Wang *et al.* [236] propose a distributed scheme with explicit dynamic data support (including block update, delete, and append) to achieve cloud data integrity and availability. They rely on erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. [139] leverages on the Tor architecture to provide data ownership privacy inside cloud. A system parameter controls both the degree of anonymity and the computational overhead imposed by the system.

The most widely used strategies for the anonymization of data content are differential privacy [91] and k-anonymity for privacy preserving microdata release [208]. These techniques are used as a preprocessing step to anonymize private data content before their submission to the cloud [139]. The proposals presented in [72, 76] take advantage of group signature scheme as anonymous access method. The first definition of group signatures was proposed by Chaum in [72]. This kind of signatures is defined as a “generalization” of the credential/membership authentication schemes, in which one person proves that he belongs to a certain group. The authors of [76] implement SPICE, a digital identity management system applicable to cloud environment, which combines two group signatures to make the same signature look different for multiple uses. The main drawback of group signature scheme is that the signature size grows with the number of user, thus making these approaches inefficient in many application contexts.

A recent proposal presented in [205] describes a decentralized access control technique with anonymous authentication, which provides user revocation and prevents replay attacks. The limitation of such an approach is that the different requests of a single user in a session could be linked together, thus resulting in a behavioral-based attack. In contrast, our technique is able to protect users also against such a type of attack.

8

Security of Transactions: Tweetchain

Blockchain technology allows mutually distrustful parties to transact safely without trusted third parties and avoiding high legal and transactional costs. Despite the rapid grow of interest of both researchers and companies in Blockchains, it is well-known that the protocol (even in the 2.0 version) has some weaknesses. In this chapter, we propose an alternative public ledger that, instead of the P2P network and the Blockchain protocol, leverages the popular social network Twitter, by building a meshed chain of tweets to ensure transaction security. Importantly, Twitter plays neither the role of trusted third party nor the role of ledger provider. From a conceptual point of view, the protocol is fully decentralized as in Blockchain, but the weaknesses above mentioned are overcome.

8.1 Background

Blockchain, originally published by Satoshi Nakamoto in [177], is known in the scientific literature as the main technical innovation of Bitcoin. In [177] the author presents Bitcoin electronic payment system as a revolutionary technology based on cryptographic proof instead of trust. Indeed, replacing the central server signature with a consensus mechanism based on proof of work, Bitcoin allows any two willing entities to transact directly with each other without relying on a trusted third party.

Every user who wants to adopt this technology has to store locally on the computer hard drive the Blockchain, in order to run a full version of the Bitcoin software. Blockchain assumes the function of a distributed ledger recording the history of every transaction sent and confirmed on the Bitcoin Network.

Specifically, a Blockchain is a distributed database that holds a continuously-growing list of blocks. The single block contains timestamped batches of valid transactions within a certain period of time, and a hash of the previous block. All these hashes link the blocks together in linear and chronological order and are used to assure that all blocks are well formed and not tampered with.

Figure 8.1 shows the representation of a Blockchain.

Within a block, there is a list of transactions created by participants any time they want to exchange a cryptocurrency (in the case of Bitcoin). Even transactions are linked together in a sort of chain. This chain is valid if it is composed of valid transactions *(i)* digitally signed, *(ii)*

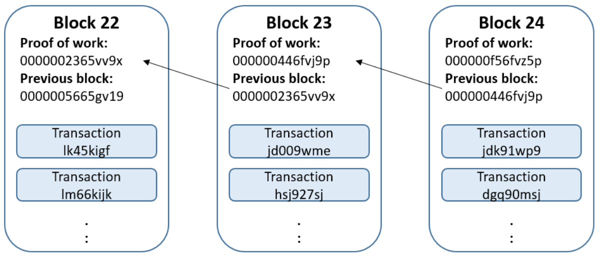


Fig. 8.1: A scheme representing the Blockchain structure.

that spend one or more unspent outputs of previous transactions, *(iii)* and for which the sum of transaction outputs must not exceed the sum of inputs.

Another feature of Blockchain technology is that it is public and decentralized. Indeed, the Blockchain runs on a widespread network of computers, called *miners*, holding all data in the Blockchain, and working on expanding it. Their function is to compete in order to form new blocks, that can be added or not to the Blockchain according to a consensus schemes, also known as *proof of work*.

This algorithm dictates the way all nodes can agree that a miner includes its block into the Blockchain. This is done establishing if the miner has done a certain amount of work that entitles it to the block reward (e.g., the insertion of its block in the Blockchain). In the case of Bitcoin, the task to accomplish for the miner is to compute a string, that when concatenated with the hash of the previous block header and then hashed, returns a string with a certain amount of preceding zeroes.

More in detail, after a new transaction *tr* is created, the network executes the following steps:

1. *tr* is broadcast to all nodes.
2. Each node collects *tr* into a block.
3. Each node works on finding a difficult proof of work for its block (e.g., it runs the algorithm described above).
4. Once the proof of work is found, the node broadcasts the block to all nodes.
5. Nodes accept the block only if all transactions in it are valid and not already spent. To validate a block, a node only needs to keep a copy of the block headers of the longest proof of work chain (which the node can get by querying network nodes) and obtain the Merkle branch linking the transaction to the block it is timestamped in. By linking the transaction to a place in the chain, the node can see if another network node has accepted it, and blocks, added after it, further confirm the network has accepted it.
6. Nodes use the hash of the accepted block as the previous hash for the following blocks.

The longest chain is always considered the correct one and all nodes will keep working on extending it. If two different versions of the next block are broadcast in the same time, nodes may receive one or the other first. In that case, they consider the first one they received, saving the other branch, in case it becomes longer. When the next proof of work is found and one branch becomes longer the tie will be broken, and all the nodes working on the other branch will then switch to the longer one.

Even though a node does not receive a block, it can request it when it receives the next block and checks whether it missed one. Thanks to that, the network is tolerant to dropped messages and new transactions, once broadcast, need to reach many nodes, but not the whole network.

As long as one single entity does not hold majority of the computing power, this algorithm can mathematically ensure security on the block chain. Indeed, the fact that the proof of work is made starting from the hashed header from the previous block, and that headers contain a hash of all transactions in that block (as illustrate in Figure 8.1), changing an old transaction requires that an attacker computes again the proof of work for all subsequent blocks. Moreover, after that he has to continuously add blocks to this chain at a higher rate than the legitimate chain to make the system using this chain instead of the right one. Indeed, this attack result infeasible unless the attacker does not hold the most of the network.

Moreover, the growing number of miners running a node on their hardware increases the security of the system. Therefore, Blockchain protocol adds an incentive to reward nodes that keep mining and use their processing power (in terms of CPU time and electricity consumptions), to create blocks. In case of Bitcoins, the first transaction in a block is a special transaction that starts a new coin owned by the miner that create the block. The incentive can also be funded with transaction fees. If the output value of a transaction is less than its input value, the difference is a transaction fee that is added to the incentive value of the block containing the transaction.

Moreover, privacy can still be guaranteed by keeping public keys anonymous. In this way the public can see that someone is sending an amount to someone else, but without linking the transaction to anyone.

8.2 The Tweetchain Model

In this section, we describe the model underlying our proposal.

The main entities of the model are:

- The Twitter social network, and, in particular, the followings among features:
  1. the posting of *tweets* for registered users;
  2. the notification on the *follows* activity;
  3. the searching for information by *hashtags*.
- a welcome profile W used to implement a sort of yellow page support.
- the `Tweetchain` community, namely C, of users who join the `Tweetchain` protocol.

8.3 Basic Approach

In this section, we describe our approach and the structure of the messages exchanged by the involved entities.

As a prerequisite, to participate in the `Tweetchain` community, a user must be able to build, by starting from a secret, a (SHA-256 based) hash chain of a given size, say k, which will be used to maintain all his timeline activities linked together. This way, as will be clearer in the following, no modification can be done on older messages without compromising the remaining part of the user timeline. The value k, representing the length of the hash chain, is a system parameter which also limits the maximum size of the chunk of the user timeline (intended as number of tweets) that must be consider to verify the validity of a given message.

All the detail on the usage of this hash chain will be clarified in the following. As a further observation, we will consider our system in a steady-state, meaning that there are always at least $s = \frac{2t}{1-m}$ members in the `Tweetchain` community, where t and m are system parameters discussed in Section 8.4.

Now, we are ready to describe our proposal. The `Tweetchain` paradigm is composed of the following protocols.

Registration. It is executed by each user, say x, who wants to become member of the `Tweetchain` community C. Clearly, a prerequisite of this protocol is the sign up to Twitter in order to create a profile on it.

The first step he performs is to follow the welcome profile W and to publish a `hello` tweet with the following structure:

$$\langle\ \#HC_x^1\ \ \#HC_W^1\ \ \texttt{Hello}\ \ @W\ \rangle$$

where $\#HC_x^1$ and $\#HC_W^1$ are Twitter hashtags with the base64 encoding of the first element of the hash chain of x and W as text, respectively, and $@W$ is a Twitter reference to the welcome page W.

After that, W verifies the tweet of x and sends a confirmation tweet as a `welcome` message with the reference to this user and a link to his `hello` tweet. Suppose that W has already posted $i-1$ tweets, then the `welcome` message for x will have the following structure:

$$\langle\ \#HC_W^i\ \ \texttt{Welcome}\ \ @x\ \ \#HC_x^1\ \ \#TID_x^1\ \rangle$$

where $\#HC_W^i$ and $\#HC_x^1$ are Twitter hashtags of the base64 encoding of the ith element of the hash chain of W and the first element of the hash chain of x, respectively, $@x$ is a Twitter reference to the user x, and $\#TID_x^1$ is a Twitter hashtag with the ID of the first tweet (`hello`) of x as text. Observe that, the ID of a tweet (or status ID) is always unique inside Twitter.

As a consequence, W contains at least one tweet for each member of the community in join-chronological order. After this, x generates at random the set F_x of s-followings who will validate his transactions in the future. This set is built as follow:

- x retrieves his Twitter identifier. (Recall that each Twitter user has a unique 64-bit numeric identifier.)
- Then, the identifier is used as seed to a community-known PRNG to extract s random numbers and for each number, say n, computes $n \bmod w$. Here w is the total number of tweets posted by W (i.e., the size of the `Tweetchain` community).
- At this point, the numbers computed above are used as indexes to select s distinct screen names from the welcome profile W.

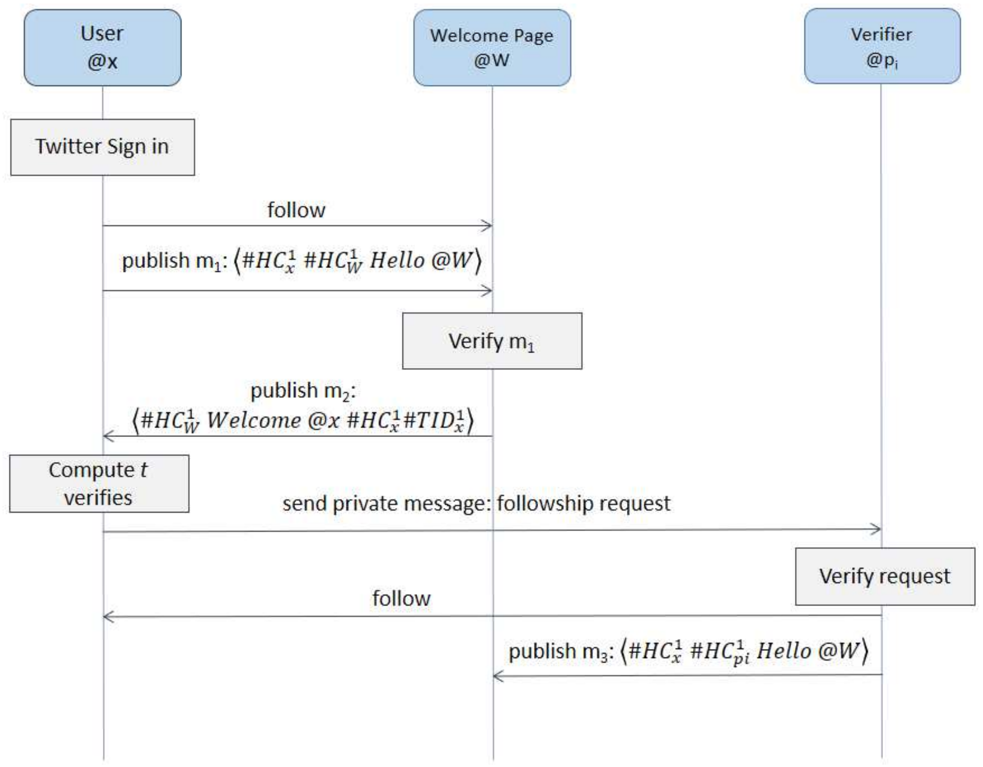

User
@x
Welcome Page
@W
Verifier
@p$_i$
Twitter Sign in
follow
publish m$_1$: $\langle \#HC_x^1\ \#HC_W^1\ Hello\ @W \rangle$
Verify m$_1$
publish m$_2$:
$\langle \#HC_W^1\ Welcome\ @x\ \#HC_x^1 \#TID_x^1 \rangle$
Compute t verifies
send private message: followship request
Verify request
follow
publish m$_3$: $\langle \#HC_x^1\ \#HC_{pi}^1\ Hello\ @W \rangle$

Fig. 8.2: A scheme representing the registration procedure.

- x sends a *private* message to each of the s profiles, whose screen names have been derived in the previous step, to ask them to follow him.
- After verifying the legitimacy of the request of x by using the community PRNG, each of the profiles contacted by x adds a follow link towards x and duplicates the `welcome` tweet of W by replacing $\#HC_W^i$ with their current hash chain element.

Transaction generation.

This protocol allows the generation of a new transaction. Similarly to what happens in Blockchain, each transaction carries different information, such as:

- The timestamp of the generation.
- A content, i.e., the transaction payload.
- An input transaction.
- A target profile acting as transaction recipient.

In our protocol, the generation of a new transaction is assimilated with that of a new *tweet* by the user, in the following referred as *t-tweet*. According to the requirements described above, the i-th *t-tweet* of the user x, will have the following structure:

$$\langle\ \#HC_x^i\ \ \#TID_y^p\ \ \texttt{content}\ \ @r\ \rangle$$

Where $\#HC_x^i$ is a hashtag with the base64 encoding of the ith element of the hash chain of x, $\#TID_y^p$ is a hashtag of the ID of the p-th tweet posted by the user y and used as input for this transaction, and $@r$ is a Twitter reference to the recipient r of this transaction.

Algorithm 5 Registration Protocol

```
Notation  $C$: the Tweetchain community;
    $x$: a Twitter user who wants to join $C$;
    $W$: the welcome page;
    $F_x$: the set of $s$-verifiers of $x$;
    $n$: a random number.
Procedure  Registration to $C$
 1: $x$ follows $W$
 2: $x$ publishes $\langle$ $\#HC_x^1$ $\#HC_W^1$ Hello @$W$ $\rangle$
 3: $W$ verifies $\langle$ $\#HC_x^1$ $\#HC_W^1$ Hello @$W$ $\rangle$
 4: if Verification is ok then
 5:     $W$ publishes a confirmation tweet: $\langle$ $\#HC_W^i$ Welcome @$x$ $\#HC_x^1$ $\#TID_x^1$ $\rangle$
 6:     $x$ retrieves his Twitter identifier
 7:     $x$ extracts a set $N$ of $s$ random numbers trough a PRNG known by $C$
 8:     for all $n \in N$ do
 9:         $x$ computes $v_n = (n \; mod \; w)$
10:         $x$ retrieves the screen names corresponding to $v_n$ in $W$ and adds it to $F_x$
11:     end for
12:     for all $i \in F_x$ do
13:         $x$ sends a private message to $i$
14:         $i$ verifies the request of $x$
15:         if Verification is ok then
16:             $i$ follows $x$
17:             $i$ publishes $\langle$ $\#HC_I^j$ Welcome @$x$ $\#HC_x^1$ $\#TID_x^1$ $\rangle$
18:             close
19:         end if
20:     end for
21:     close
22: end if
```

As soon as x posts a new *t-tweet*, the s users of F_x following him will be notified by the Twitter platform automatically. They will proceed by verifying the legitimacy of this new transaction by using the verification protocol described below. After running the verification procedure, they will publish a confirmation tweet on their timeline. Now, let v be one of the users of F_x and let $j-1$ be the number of tweets generated by v till now, his confirmation tweet for the transaction of x will be:

$$\langle \; \#HC_v^j \;\; @x \;\; \#TD_x^i \;\; \texttt{status} \;\; \#TID_y^p \;\; \texttt{content} \;\; @r \; \rangle$$

Here $\#HC_v^j$ is the hashtag of the j-th element of the hash chain of the verifier v, $@x$ is the reference to the user x, $\#TD_x^i$ is the hashtag of the ID of the *t-tweet* generated by x, and *status* can either be 1 for *success* or 0 for *failure* on the basis of the verification result. The remaining of this tweet is the essential part of the body of the *t-tweet* of x necessary to reconstruct the original tweet in case of deletion done by x.

Verification. This protocol is used to check the validity of a transaction. Now, suppose a verifier, say v, wants to verify the i-th transaction of the user x having the p-th transaction of a user y as input and the user r as target. The transaction of x will have following structure:

$$\langle \; \#HC_x^i \;\; \#TID_y^p \;\; \texttt{content} \;\; @r \; \rangle$$

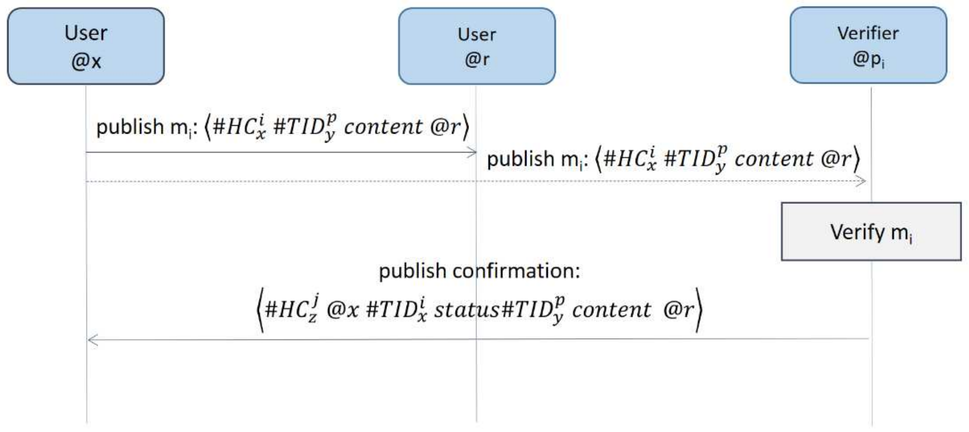

User @x
User @r
Verifier @p$_i$
publish m$_i$: $\langle \#HC_x^i \ \#TID_y^p \ content \ @r \rangle$
publish m$_i$: $\langle \#HC_x^i \ \#TID_y^p \ content \ @r \rangle$
Verify m$_i$
publish confirmation:
$\langle \#HC_z^j \ @x \ \#TID_x^i \ status\#TID_y^p \ content \ @r \rangle$

Fig. 8.3: A scheme representing the transaction generation.

The protocol works by verifying each part of this *t-tweet*. Observe that, concerning the verification of the content, this is not considered here as it is strictly related to the objective of the transaction, which is not specified in this chapter and, therefore, it is fully application dependent. As for the verification of $\#HC_x^i$, first v checks whether this hash chain element has been already used, in this case the verification will fail, otherwise the verifier has to compute the SHA-256 hash of $\#HC_x^i$. Due to the hash chain property, the results of this computation should be $\#HC_x^{i-1}$. Therefore, a search on Twitter for $\#HC_x^{i-1}$ should return the previous tweet posted by x. The goal of the verifier is to find the previous *t-tweet* of x or the initial `hello` message. Now, because x will also post confirmation tweets for other users' transactions, in the case $\#HC_x^{i-1}$ refers to a confirmation tweet, the procedure above is repeated until either a *t-tweet* or the `hello` tweet is found. Let $\#TID_x^{i-1}$ be the ID of such a tweet, the verifier has now to check whether he has confirmed this tweet in the past (i.e., has posted a confirmation tweet with status 1 corresponding to it). If this is not the case, then the verifier will not confirm ($status$ 0) the new transaction, otherwise he will proceed with the verification of the second part of the new *t-tweet* of x. The verification of $\#TID_y^p$ implies the verification of the validity of the input transaction. As said above, the input is the p-th *t-tweet* of the user y. The validity of this tweet is related to the presence of at least t confirmation tweets among the s generated by the verifiers associated with y. Because each confirmation tweet contains the ID of the corresponding *t-tweet*, a search in Twitter for $\#TID_y^p$ will return all the confirmation tweets for the p-th *t-tweet* of p. Now, the verifier has to check both the presence of t confirmations with status 1 and that they have been posted by the legitimate verifiers for y. Moreover, v checks whether the target of $\#TID_y^p$ is x and that it has not been already used as input in any other transaction. The last verification done is the check of the existence of r (i.e., the target user of this new transaction) in the `Tweetchain` community. This is obtained by searching for $@r$ in Twitter and by verifying the presence of the `welcome` message in W along with t confirmations of the verifiers of r. At end of these steps, if all the verifications succeeded, then the verifier will post a confirmation tweet with status 1 for the new *t-tweet* of x, otherwise the status of this confirmation will be set to 0.

Algorithm 6 Generation and Verification Protocol

```
Notation  $C$: the Tweetchain community;
    $x$: a Twitter user who generate a t-tweet;
    $F_x$: the set of $s$-verifiers of $x$;
    $v$: a verifier $\in F_x$;
    $h$: a 256-bit cryptographic hash function.
Procedure  t-tweet Generation
 1: $x$ publishes $\langle$ #$HC_x^i$ #$TID_y^p$ content @$r$ $\rangle$
Procedure  t-tweet Verification performed by $v$
 2: if #$HC_x^i$ has been already used then
 3:     $v$ publishes $\langle$ #$HC_z^j$ @$x$ #$TD_x^i$ 0 #$TID_y^p$ content @$r$ $\rangle$
 4:     close
 5: end if
 6: $v$ computes $h$(#$HC_x^i$) = #$HC_x^{i-1}$
 7: $v$ searches for #$HC_x^{i-1}$
 8: while t-tweet OR hello tweet is found do
 9:     $v$ searches for #$HC_x^{i-1}$
10: end while
11: if #$HC_x^{i-1}$ is not found then
12:     $v$ publishes $\langle$ #$HC_z^j$ @$x$ #$TD_x^i$ 0 #$TID_y^p$ content @$r$ $\rangle$
13:     close
14: end if
15: $v$ retrieves #$TID_x^{i-1}$
16: if $v$ has not already confirmed this t-tweet then
17:     $v$ publishes $\langle$ #$HC_z^j$ @$x$ #$TD_x^i$ 0 #$TID_y^p$ content @$r$ $\rangle$
18:     close
19: end if
20: $v$ searches for #$TID_y^p$
21: if $v$ finds less than $t$ confirmation tweets OR the target of #$TID_y^p \neq$ @$x$ OR @$x$ has been already used as input in
    any other t-tweet then
22:     $v$ publishes $\langle$ #$HC_z^j$ @$x$ #$TD_x^i$ 0 #$TID_y^p$ content @$r$ $\rangle$
23:     close
24: end if
25: $v$ searches for @$r$ in $C$
26: if $v$ does not finds a welcome message for @$r$ in $W$ OR less than $t$ confirmation tweets of the verifiers of $r$ then
27:     $v$ publishes $\langle$ #$HC_z^j$ @$x$ #$TD_x^i$ 0 #$TID_y^p$ content @$r$ $\rangle$
28:     close
29: end if
30: $v$ publishes $\langle$ #$HC_z^j$ @$x$ #$TD_x^i$ 1 #$TID_y^p$ content @$r$ $\rangle$
```

8.4 Security analysis

In this section, we describe a security analysis of our approach showing that it accomplishes its objectives also in presence of attacks. Therefore, in the following, we describe the security model and analyze the security properties of our proposal.

To analyze the security properties, first our threat model includes the following assumptions:

A1 The attacker cannot add or compromise information shown on the social network accounts of any of the users of the Tweetchain community.

A2 The Twitter service is up and running as expressed in its use conditions and does not intend to block the Tweetchain community (i.e., we assume that Twitter only could block single individuals, not the whole community).

A3 Collision, preimage and second preimage attacks on the cryptographic hash function are infeasible.

A4 The attacker cannot know the secrets used by users to generate their hash chains.

A5 Given t verifiers the maximum number of those who do not respond according to the protocol to the verification request is $m \cdot t$ where $0 \leq m < 1$.

A6 At most t user can collude to to break security properties of the protocol.

The last assumption merits a brief comment. We recall that, in our protocol, confirmations are produced collaboratively by several users playing as verifier. Some users might be corrupted by an adversary, but we assume an honest majority of users at all times. This is a common assumption borrowed by the field of e-voting [256, 100], As a consequence, our technique is parametric with respect to the value t. It is chosen in such a way that the likelihood that t randomly selected users misbehave is negligible. Similar considerations can be done for Assumption **A5**.

Now, we are ready to identify the security properties (hereafter, **SP**) that our system has to assure and discuss the attacks and the countermeasures to contrast them.

SP1- Transaction Authenticity. A transaction and the user generating it can always be verified by the Tweetchain community.

Attack AA1: *An adversary tries to impersonate the welcome profile W to tamper the list of verifiers for a user.*
This attack is contrasted by the fact that the screen name of each account is unique in Twitter and that associated with W is known to all the Tweetchain community. Therefore, due to Assumptions **A1** and **A2** this attack cannot happen.
Attack AA2: *An adversary creates multiple accounts and tries to use them as verifiers of his own transactions.*
This attack is contrasted by two security mechanisms introduced in our approach: (i) the welcome profile yellow page service; (ii) the PRNG-based mechanism for verifiers assignment.
The former forces each user who wants to join the Tweetchain community to perform a preliminary registration to W. After this, the latter security mechanism ensures that the election of verifiers for a given user is publicly verifiable as the PRNG-base strategy is entirely reproducible by each user in the Tweetchain community. Therefore, also thanks to Assumption **A1**, this attack cannot happen.

SP2 - Transaction Integrity. The whole message (*t-tweet*) representing a transaction cannot be tampered once posted on the system.

Attack AI1: *Some of the verifiers do not execute the verification protocol invalidating a transaction.*

Thanks to Assumption **A5** we know that, among t verifiers, the maximum number of those who may not collaborate properly by posting the confirmation tweet is $m \cdot t$, where $0 \leq m < 1$. To resist such attack we have to set the number of verifiers for any user of the community to $\frac{t}{1-m}$. Indeed, our aim is to obtain at least t confirmations to validate a new transaction. We know that if we consider an initial set of t verifiers, to obtain t confirmations we have to add $(m \cdot t)$ verifiers to this set, to compensate those who may not respond among the first t. Now, in the set of $(m \cdot t)$ added users, once again, we have that $m \cdot (m \cdot t)$ may not respond. Therefore to obtain at least t confirmations we need to set the number of verifiers to $t + (m \cdot t) + (m^2 \cdot t) + \cdots + (m^i \cdot t) = t \cdot \sum_{i=0}^{\infty} m^i = \frac{t}{(1-m)}$. With this setting, such attack is contrasted.

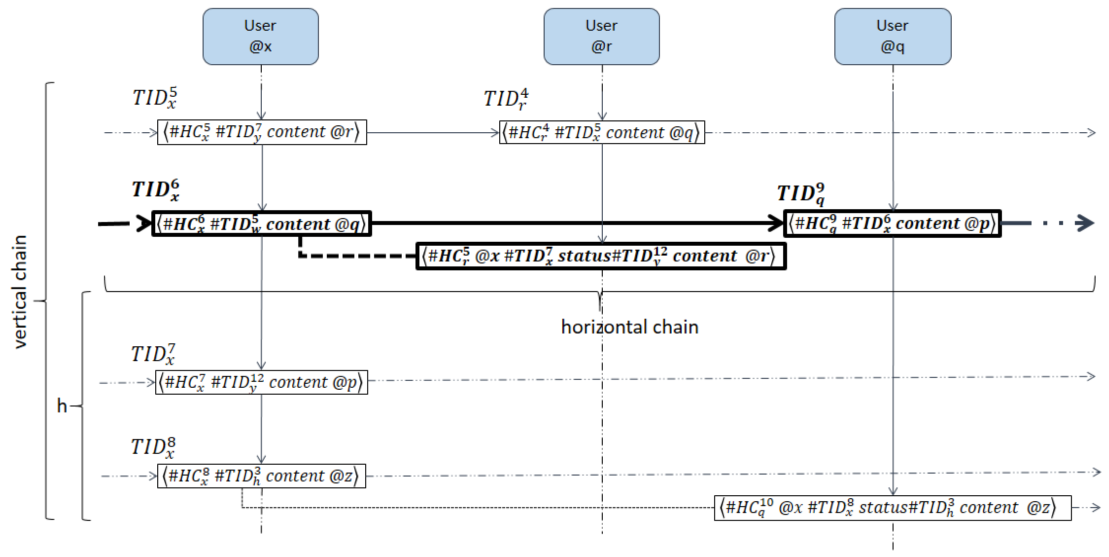
Attack AI2: *Some of the verifiers collude to compromise the integrity of a transaction.*

This attack is contrasted by considering that, according to Assumption **A6**, at most t users can collude to perform an attack. Clearly, if we want to obtain t confirmations to validate a new transaction, to contrast the collusion attack, we should set the number of verifiers for each user to $2 \cdot t$. By combining this intuition with the countermeasure adopted to make the approach resistant to Attack **AI1** we have that the number of verifiers for each user has to be set to $\frac{2 \cdot t}{(1-m)}$.

Attack AI3: *Twitter, playing as an adversary, tries to compromise the integrity of a transaction by deleting a piece of the chain (even the whole) this transaction depends on.*

Suppose the adversary has as target a transaction q. The logic underlying this attack is that if the adversary is able to delete a partial or the whole set of transactions and corresponding confirmations related to q, the integrity of the latter is compromised as the verification of its input would fail. However, to prevent such an attack in our approach not only each transaction is linked in a chain of transactions posted by other users (horizontal chaining) but it is also linked, by means of the user hash chain, to messages generated by its owner before and after it (vertical chaining). Moreover, this reasoning applies also for the confirmations of each involved transaction, thus creating a mesh of chained transactions and confirmations around the attacker target, i.e. q. The deletion of any piece of this scheme will produce a domino effect causing all the transactions in this mesh (possibly extending to the entire community due to the small world property [141]) to be compromised. This would intrinsically go against to what provided by Assumption **A2**.

Clearly, the reasoning above holds if there are transactions in the network which are more recent than the piece of chain the adversary is planning to delete. Consider the improbable situation in which all the transactions involved in the piece of the horizontal chain under attack are the last posted by the corresponding users. In this case, to contrast such an attack, we enforce that a transaction, already validated through the verification protocol, can be considered in a *safe* state only if there are at least h transactions generated after any element of the horizontal chain linking it (a similar measure is adopted also in Blockchain). This reasoning is sketched in Figure 8.4, in which the horizontal chaining, the



User @x
User @r
User @q
TID_x^5
$\langle \#HC_x^5\ \#TID_y^7\ content\ @r \rangle$
TID_r^4
$\langle \#HC_r^4\ \#TID_x^5\ content\ @q \rangle$
TID_x^6
$\langle \#HC_x^6\ \#TID_w^5\ content\ @q \rangle$
TID_q^9
$\langle \#HC_q^9\ \#TID_x^6\ content\ @p \rangle$
$\langle \#HC_r^5\ @x\ \#TID_x^7\ status\#TID_y^{12}\ content\ @r \rangle$
vertical chain
horizontal chain
h
TID_x^7
$\langle \#HC_x^7\ \#TID_y^{12}\ content\ @p \rangle$
TID_x^8
$\langle \#HC_x^8\ \#TID_h^3\ content\ @z \rangle$
$\langle \#HC_q^{10}\ @x\ \#TID_x^8\ status\#TID_h^3\ content\ @z \rangle$

Fig. 8.4: A fragment of the grid structure created by our protocol.

vertical chaining, and the depth h (necessary to consider the *safe* state for a transaction) are highlighted.

SP3 - No repudiation of Transactions. The user generating a transaction cannot repudiate it.

Attack AR1: *An adversary tries to make ambiguous a transaction by forging another one with the same input transaction.*

In this case, the attacker tries to create a fork in the chain using the same input transaction but with different recipient. However, this attack cannot occur due to the verification mechanism according to which each verifier has to check for the presence of the input in a previous transaction (see Section 8.2). Therefore, also thanks to Assumption **A1**, the attempt of creating a fork is demised because the confirmations will have *status* 0.

Attack AR2: *The user, acting as an adversary, tries to repudiate a transaction by deleting the corresponding t-tweet.*

This attack can be performed in two ways: *(i)* the adversary deletes the *t-tweet* before all the confirmations are produced; *(ii)* the adversary deletes the *t-tweet* after the confirmations. In the first case, the transaction will be considered not valid by the verification protocol and hence, it will be not usable as input for future transactions. Therefore, no real advantage is produced with this attack. In the second case, the adversary tries to make the transaction valid by waiting for all the confirmations to be produced, and then proceeds by deleting the original *t-tweet* with the objective of repudiating the transaction itself. However, the attack fails as, according to the verification protocol and due to Assumptions **A5** and **A6**, the transaction will be considered valid because at least t confirmations will be found.

8.5 Discussion

The assurance of information is becoming an issue more and more important in the current era. Moreover, an emerging trend is to do not rely on a single body, to ensure scalability and to avoid to require the (sometimes unrealistic) trustworthiness of this body. Blockchain is the other extreme, because transactions are broadcasted to the entire community and (potentially) all the participants compete to verify and approve transactions. As we have commented earlier, this *global* competition results in a huge amount of work and energy loss. Mainly for this reason, an extensive use of Blockchain could be inopportune. However, a great attention towards Blockchain has been recently devoted by both researchers and companies, due to its high applicative power, mainly relying on the idea that Blockchain can implement a public, shared ledger without dedicating any trusted entity.

The aim of our research is to find an alternate to Blockchain that can favor the massively extension of public-ledger-based applications, also by solving the main drawback of Blockchain that is the need of the proof of work. We thought that this goal can be reached by exploiting the native power of online social networks to connect people and playing *for free* as a platform for massive collaboration. As a matter of fact, so far social networks have been mostly used as communication media. We think they can become part of people workflows, with reciprocal advantage: for people high usability, low cost, high availability, for social network providers more central role in the society. Conversely, the P2P approach required by Blockchain is certainly less universally accepted, results in more client-side invasivity, enforces a stratification of users making only the *oligarchy* of full nodes the real participants in the community (the majority of nodes must trust full nodes). Moreover, for full nodes, a great effort in terms of storage space, bandwidth, and computational power, is required.

The solution we have proposed, called `Tweetchain`, solves the above problems. We summarize here its main features.

- As Blockchain, `Tweetchain` is a platform for different applications. In other words, the content of the transactions is someway independent from the protocol. The mechanism of resources referred by tweets, may allows us to use the protocol for many applications.
- `Tweetchain` ensures, if required, the same anonymity degree as Blockchain. Instead of different public keys, the user may use different Twitter profiles and accessing the network by TOR to obtain privacy. Concerning the possibility for an attacker to break this privacy, it is easy to see that `Tweetchain` has the same resistance as Blockchain.
- Twitter does not play the role of trusted third party. Moreover, it does not provides specific functionalities of the protocol. It just publishes the tweets of the protocol exactly as the other ones. As shown in the security analysis, Twitter cannot succeed as adversary, only could stop the entire community. Even though this occurrence can be considered very improbable in a real-life massive utilization of the protocol, it is easy to understand that our approach can be thought at a multiple-social-network level, over the hundreds of existing social networks. This way, the probability that the the community can be stopped becomes negligible (even though only the biggest few social networks are involved).

- In `Tweetchain`, there is no need of proof of work. Therefore, there is no lost work in the community. The confirmation of the transactions is done by a number of participants ensuring that the probability of collusion is negligible. Thus, between the centralized scenario of *one is the trustee* and the other extreme aimed (but not obtained) by Blockchain of *all are trustees*, we find a sort of optimum equilibrium. Moreover, all the participants have the same rights and exchange the same roles, in a truly conceptual P2P fashion.
- No P2P application must be installed by users. They just are required to run an application accessing their Twitter profile. This appears more efficient and we expect that it would be more acceptable because perceived as more secure and more transparent than P2P. Besides the perception, it is rather intuitive that enabling P2P functionalities may enlarge the attack surface client-side. Concerning the access control of the application to the Twitter profile, approaches like [47] can be used to prevent application misbehaving.
- If we want to use `Tweetchain` for those applications in which the identity of actors is a critical factor, we can enforce the use of only verified Twitter profiles with double factor strong authentication [8] or interfacing our protocol to some public digital identity system.
- As for Blockchain 1.0, a lot of possible applications can be implemented over `Tweetchain`. We do not consider that `Tweetchain` should be proposed as crypto-currency like Bitcoin. This because the success of a crypto-currencies depends on a lot of factors that we did not analyzed in this thesis. However, many other applications can be thought, related to voting, e-commerce, document exchanges, ticketing, crowdshipping, reputation systems, tourism, advertising, etc. As a future work, beside a full implementation, we plan to implement on `Tweetchain` also the paradigm of smart contracts, thus enabling `Tweetchain` to supports all the applications runnable over Blockchain 2.0.

8.6 Related work

Since its creation in 2008 [177], Blockchain technology has sparked a growing interest within the scientific community.

Indeed, many works try to solve problems related to Blockchain in order to improve its functionalities. In particular, in [28] the authors propose a new technology, called pegged sidechains, which enables Bitcoins and other ledger assets to be transferred between multiple Blockchains. This allows the interoperation among different cryptocurrency systems and gives users access to these systems reusing Bitcoin currency and maintaining the assets they already own.

In [143] an approach to guarantee privacy over smart contract systems based on decentralized cryptocurrencies is presented. In the classical Blockchain protocol, the entire sequence of transactions taken in a smart contract are propagated across the network and saved in the Blockchain, and therefore are publicly visible. This system does not store financial transactions in plain text on the Blockchain, solving also deanonymization attacks.

We do not consider the problem of privacy of transactions in our work. By the way, as also done in Blockchain, by using anonymous Twitter profiles in conjunction with some anonymous communication service, such as Tor [89], a user can certainly obtain these privacy requirements.

Whereas, in [97] the authors propose Bitcoin-NG a Byzantine fault tolerant protocol in order to solve the intrinsic scalability limits of Blockchain. Our consensus model can be framed within the family of state-machine replication (SMR) protocols that consists essentially in replicating deterministic state machines in different hosts [211]. Moreover, it guarantees consensus despite participation of malicious (Byzantine) nodes because all the transaction exchanged are published on Twitter [21].

Always in this direction, the works presented in [158, 216] describe alternate structures to the chain in order to reach higher rates (e.g., a directed acyclic graph of blocks and a tree of blocks, respectively). Our work maintains a meshed chain structure, thus forming a sort of grid (as shown in Section 8.4). Moreover, the block entity is assimilated with a single transaction.

Also performing transactions off the chain can be seen as an alternative to improving the bandwidth and latency [83, 195]. In our protocol we do not need that verifiers are always all online and fully operational (see Sections 8.2 and 8.4). Moreover, once a verifier signs in Twitter he is prompted to produce his confirmations before he can proceed with his tasks.

More than one authors suggest a mechanism based on incentive to motivate transaction propagation, deterring the formation of large open mining pools [96, 157]. Within our protocol we do not consider incentives. Indeed, it is made up peer nodes, assuming in turn the role of verifiers and transaction generators. If a node wants to benefit `Tweetchain` services it has to agree with the protocol and serve as verifier when needed.

9

Urban Security and Third-Party Cloud: the case of Video Surveillance

Cloud computing provides users with the possibility to store their data in third-party servers. These data centers may be untrusted or susceptible to attacks, hence they could return compromised query results once interrogated. This issue becomes particularly crucial in the case a third-party cloud is used to store video-surveillance data. Query integrity has been widely investigated in the literature, and a number of methods have been proposed to allow users to verify that query results are complete *(i.e., no qualifying tuples are omitted),* fresh *(i.e., the newest version of the results are returned), and* correct *(i.e., the result values are not corrupted). However, in an application scenario, in which append operations and range queries on stored data streams are dominant, and efficiency is a critical factor, classical techniques for query integrity appear little suitable. This chapter proposes a new solution to overcome these drawbacks.*

9.1 Background

Understanding the following concepts and cryptographic primitives is useful for the comprehension of our approach and some related works we cite in this chapter:

- A **Hash chain** is given by the successive application of a cryptographic hash function H to a data. For instance, a hash chain of length 3 can be obtained by applying three times the hash function H to a string x $H(H(H(x)))$.
- A **Hash list** is a list of hashes in which each hash value is referred to a different data block. It is used to guarantee data integrity and it is better than a simple hash of the entire file because, in the case of damage of a single block, this can be identified. Often, a top hash is computed (i.e., a hash of the whole hash list) to make a first quick check on the entire data.
- A **Merkle hash tree** [174] is a more advanced form of hash list which allows verification of the contents of large data structures. It is defined as a tree in which every node (except for the leaves) is labelled with the hash of the labels (or values, in case of leaves) of its children nodes. An example of Merkle hash tree is presented in Figure 9.1. $H_{0,0}$ and $H_{0,1}$ are the hash values of the first and the second data blocks D_0 and D_1, respectively. Iteratively, H_0 can be computed by $H(H_{0,0}||H_{0,1})$, where $||$ is the concatenation operator. Proving that a leaf node belongs to the given hash tree requires a computational cost proportional to

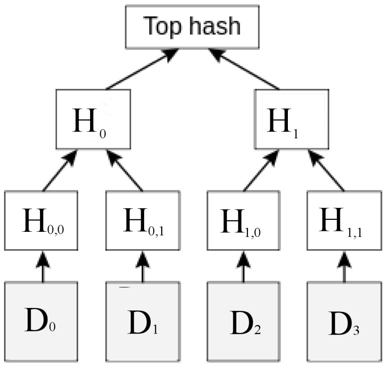


Fig. 9.1: An example of Merkle Hask Tree.

the logarithm of the number of nodes of the tree. For instance, to authenticate the first data block a user has to be provided with the hash value of the second data block ($H_{0,1}$) altogether with H_1. Since the user holds a signed value of the top hash, he checks if top hash is equal to $H(H(H(D_0)||H_{0,1})||H_1)$

- **MAC** means message authentication code and it is a short piece of information, associated with a message, used to assess the integrity and authenticity of the message itself.
- A **HMAC** is a specific kind of MAC constructed from a cryptographic hash functions. In particular, $HMAC(K,m) = H((K \oplus opad)||H((K \oplus ipad)||m))$ where K is a secret key generated from a master key, m is the message to be authenticated, $||$ denotes the operation of concatenation, $\oplus$ represents exclusive or (XOR) operation, and $opad$ and $ipad$ are two kinds of padding, namely outer and inner padding.

9.2 Scenario and Problem Formulation

In this section, we describe the referring scenario and formulate the problem we deal with. Specifically, we consider a particular application context in which the (possibly untrusted) cloud performs query processing on behalf of the data owner. The problem is guaranteeing query *integrity* (i.e., *completeness*, *correctness* and *freshness*) [209].

We consider a realistic scenario of video surveillance. We have a network of *(battery-powered) cameras* (such as, drones, micro-drones, insect spy drones, etc.) monitoring a high-size area and store images into the cloud. Beside allowing data storage, this server provides an interface to access data and to perform query processing on behalf of the data owner, who administrates and analyzes query results in accordance to specific application-related requirements. To optimize the process, cameras are equipped with a motion sensor to intelligently record event images only when motion is detected to allow power saving.

We remark that the choice of a specific application context is done with the purpose of highlighting the practical significance of our proposal, also by considering the importance that video surveillance is assuming nowadays. However, our solution is not *ad-hoc,* because it appears suitable to all cases in which the cloud stores data streams and the efficiency of

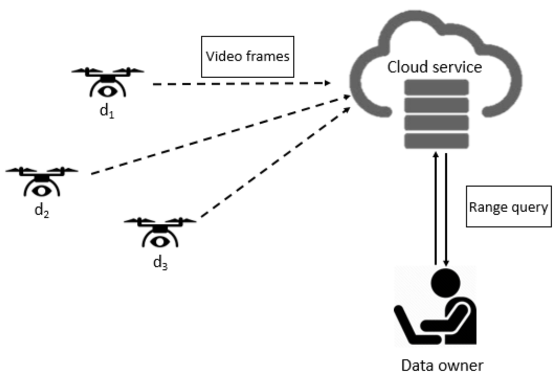


Fig. 9.2: A representation of the considered scenario.

continuous insertions is a critical aspect. Thus, many other applications can be easily found, for example in the field of Internet of Things or logistics.

In our model, we may identify the following actors:

- **The data owner:** the entity that owns the data, performs queries to the cloud server, and verifies query results.
- **The cameras:** which record video frames and store them on the cloud server.
- **The cloud server:** which hosts the database and publishes an interface for query processing.

A graphical representation of this scenario is shown in Figure 9.2.

We want to provide a mechanism allowing the verification of the completeness, correctness and freshness of query results. Clearly, battery saving assumes a very important role in this case. Therefore, one of the main problems is that excessive computation on recording sensors should be avoided. Under this assumption, as shown in detail in Section 9.4, the solutions proposed in the literature for the problem of query integrity, are little suitable. The aim of our work is to propose a more realistic (i.e., efficient) solution. We focus our attention on range queries, which ask for the images captured in the interval between two given timestamps. This type of query is the most meaningful in the considered scenario. Concerning data storing, because video frames have precise timestamps, we assume that all entities are time-synchronized and data insertion in the database is possible only if the frame timestamp is equal to the current time. As for the database dimension, we consider an aging mechanism so that only data with timestamp greater than $(t_c - t_s)$ are guaranteed in the database, where t_c is the current time.

9.3 Basic Approach

Our approach to verify query-result integrity can be classified as a deterministic technique [209]. This type of approach makes use of authenticated data structure and allows the verification of a query result leveraging verification objects that should be included in the results.

Clearly, deterministic approaches work for queries in which the conditions apply on the attribute for which the support structure has been created. Commonly, these solutions adopt hash-based tree structures (such as Merkle hash trees or MB-trees [184, 159]) to organize data. As we will analyze in Section 9.4, these structures are extremely efficient during the verification procedure, but require additional computational cost to maintain updated the structure during the insertion of new data. Unfortunately, in our application, insertions are highly frequency operations performed by those devices which have strict power consumption constraints. Indeed, query verification is performed by the data owner. As a consequence, we look for a method which supports insertions more efficiently than the hash-based tree structures.

We present our scheme in detail. We start by introducing some basic definitions.

Definition 9.1. Given a camera device s_i, we define the image sequence generated by s_i as $F^i = \{f^i_{t_1}, \ldots, f^i_{t_n}\}$, where: (*i*) $f^i_{t_j}$ is the image captured at the instant t_j for each $1 \leq j \leq n$ and (*ii*) $t_j < t_{j+1}$ (i.e., the instant t_j comes before t_{j+1} in time) for each $1 \leq j < n$.

In words, an image sequence represents a track associated with a camera device and we maintain independent sequences for different devices. Observe that, each image together with all support attributes (such as timestamp, geographic coordinates, etc.) is stored as a new tuple in the database. Therefore, throughout the paper, we will refer to a tuple as an element of an image sequence.

Definition 9.2. Given two instants t_l and t_u with $t_1 \leq t_l \leq t_u \leq t_n$, a *range query* $Q^i(t_l, t_u)$ on F^i asks for all the images $f^i_{t_x}$ such that $t_l \leq t_x \leq t_u$.

Basically, a range query is defined as a request to obtain all the images recorded by a given device s_i during the interval $[t_l, t_u]$, where t_l and t_u, with $t_l \leq t_u$, are valid timestamps during the device recording lifecycle, i.e., its total recording interval $[t_1, t_n]$.

Now, consider a device image sequence F^i. Our approach works by organizing database tuples associated with F^i in a chain. The link between two elements is built so that the owner can always verify the chain validity. Specifically, given a tuple $f^i_{t_j} = \langle a_1, \ldots, a_p \rangle$, where $a_1 = t_j$ is the tuple timestamp, and $a_2, \ldots, a_p$ are further attributes, our approach modifies it by adding an attribute encoding a link towards the next tuple in our database according to the timestamp value. Therefore, we create a new structure $\hat{f}^i_{t_j} = \langle f^i_{t_j}, MAC \rangle$, where the attribute MAC is a message authentication code and is computed by means of the function $HMAC(v, K_i)$ implementing the $HMAC$ protocol with SHA-256 as cryptographic hash function, $v = f^i_{t_j} \| e$, K_i is a secret shared by the s_i camera and the data owner, and e is either the next tuple $f^i_{t_{j+1}}$ of s_i, or a special element defined hereafter. Indeed, the chain is completed with the insertion of dummy entries representing *markers* that are used to both validate the head of the chain and to reduce the integrity verification costs by splitting F^i in time buckets. These elements are pre-added to the database and are known to all the actors involved in our scenario (i.e., they are part of the public scheme of our protocol).

Figure 9.3 shows the structure of the chain where the elements d_{T_w} and $d_{T_{w+1}}$ represent *markers*, whereas the white boxes are normal tuples.

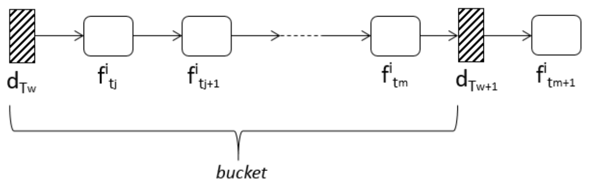


Fig. 9.3: An example of the chain of a single device image sequence.

Concerning the *markers*, they have the following basic structure: $d_{T_w} = \langle T_w, ID_b \rangle$, where T_w is the *marker* pre-fixed time (i.e., T_w is chosen by the owner during the system initialization phase), and ID_b is the bucket identifier. Clearly, as further illustrated in Figure 9.4, each *marker* has also to maintain different attributes, namely $MAC_1, \ldots, MAC_n$, representing links to devices $s_1, \ldots, s_n$, to complete the integrity chain described above. Therefore, the complete structure of *markers* will be: $\hat{d}_{T_w} = \langle T_w, ID_b, MAC_1, \ldots, MAC_n \rangle$, where MAC_i is a message authentication code associated with the device s_i and is computed by means of the function $HMAC(v^i, K_i)$ implementing the $HMAC$ protocol with SHA-256 as cryptographic hash function, $v^i = T_w \| ID_b \| e$, K_i is a secret shared by the s_i camera and the data owner, and e can be either the next s_i tuple $f^i_{t_{j+1}}$ (the first tuple in the corresponding bucket) or the next *marker* $d_{T_{w+1}}$.

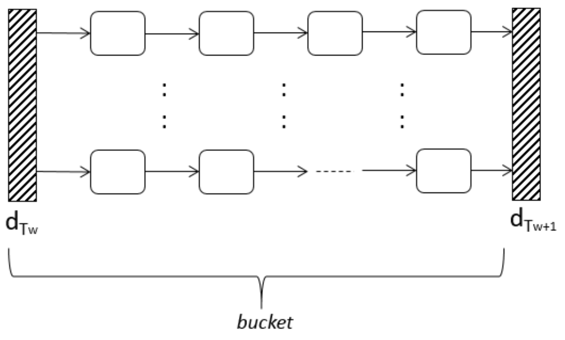


Fig. 9.4: An example of multiple device chains.

In the next sections, we will describe our protocol for the creation and population of the database, and the integrity verification mechanism.

9.3.1 Database population protocol

According to our scheme, preliminarily the data owner establishes the database life period by deciding the number of *markers* and their time position (T_w). The *marker* time positions can be simply uniformly distributed in the whole database life period or can follow specific patterns decided by the owner. For instance, he may decide to intensify the number of *markers* during specific time intervals, such as rush hours or critical daily moments. Initial *marker* values are stored in a table of the cloud database, say `m_tab`, whose content is shown in Figure 9.5. All system entities know the exact position of each *marker*. Observe that, in this initialization phase, the values of each attribute MAC_i is set to *null*.

T	ID_b	MAC_1	...	MAC_n
00:00	1	null	...	null
01:00	2	null	...	null
:	:	:	:	:

Fig. 9.5: The initial state of m_tab with *markers* positioned at every hour.

Now, we discuss how a camera s_i inserts a new tuple. The algorithm formalizing this procedure is reported in Algorithm 7: we can have two cases, depending if the table is empty or not. In the first case, no previous tuple has been inserted so the table associated with the camera sequence F^i, say s_i_tab, is empty. In this case, s_i has to link the new tuple to the *marker* with the higher time position T_w such that $t_j \geq T_w$, where t_j is the tuple timestamp. Therefore, once the right *marker* has been found, s_i will perform an update on m_tab to set the attribute $MAC_i = HMAC(d_{T_w} \| f^i_{t_j}, K_i)$ of the *marker* $\hat{d}_{T_w}$. After this, it inserts the new tuple in s_i_tab with its *MAC* attribute set to *null* and also stores it altogether with the time position T_{w+1} of the next *marker* in its local memory. The memorization of these parameters is useful for future insertions as will be clearer in the following. Observe that, we assume the data owner can always read the device-onboard memory, thus at every moment he can know the timestamp of the last tuple inserted by each device.

Consider now the second case in which s_i has to insert a new tuple in a non-empty table. Let t_j be the timestamp of the last inserted tuple and t_z be the timestamp of the tuple being inserted. Moreover let T_{w+1} be the *marker* time that s_i stored in its local memory during the previous insertion and T_{w+2} be the next *marker*. We can identify three possibilities:

1. $t_z < T_{w+1}$. In this case, the new tuple will belong to the existing bucket delimited by *markers* with time position T_w and T_{w+1}. Therefore, to maintain the chain, the tuple will be linked to the previous inserted element. To do so, the device performs an update on s_i_tab to change the *MAC* attribute of the previous inserted tuple from *null* to $HMAC(f^i_{t_j} \| f^i_{t_z}, K_i)$ and inserts the new tuple in the database. Finally, it updates the last inserted tuple in its local memory.
2. $T_{w+1} \leq t_z < T_{w+2}$. In this case, the tuple will belong to a new bucket right next to the current one; then, s_i will perform an update on s_i_tab to change the *MAC* attribute of the previous inserted tuple from *null* to $HMAC(f^i_{t_j} \| d_{T_{w+1}}, K_i)$, an update on m_tab to set the attribute $MAC_i = HMAC(d_{T_{w+1}} \| f^i_{t_z}, K_i)$ for the row corresponding to the *marker* with time position T_{w+1}. Finally, it inserts the new tuple in s_i_tab with the *MAC* attribute set to *null* and also stores it altogether with the time position T_{w+2} of the next *marker* in its local memory.
3. $t_z \geq T_{w+2}$. In this case, the tuple will belong to a new non-adjacent bucket; therefore, starting from the *marker* with time T_{w+1}, s_i has to find the *marker* with the higher time position

Algorithm 7 Data Population Protocol: Insertion of a new tuple

```
1: if $s_i$_tab is empty then
2:     search $T_w : t_j \geq T_w$
3:     update m_tab.$MAC_i = HMAC(d_{T_w} \| f^i_{t_j}, K_i)$ where $T = T_w$
4:     insert $f^i_{t_j} = \langle t_j, attr_2, \ldots, attr_p, null \rangle$ in $s_i$_tab
5:     sets $last\_tuple = f^i_{t_j}$ and $last\_time = T_{w+1}$
6: else
7:     if $t_z < T_{w+1}$ then
8:         update $s_i$_tab.$MAC_i = HMAC(f^i_{t_j} \| f^i_{t_z}, K_i)$ where $t = t_j$
9:         insert $f^i_{t_z} = \langle t_z, attr_2, \ldots, attr_p, null \rangle$ in $s_i$_tab
10:        set $last\_tuple = f^i_{t_z}$ and $last\_time = T_{w+1}$
11:    else if $T_{w+1} \leq t_z < T_{w+2}$ then
12:        update $s_i$_tab.$MAC_i = HMAC(f^i_{t_j} \| d_{T_{w+1}}, K_i)$ where $t = t_j$
13:        and m_tab.$MAC_i = HMAC(d_{T_{w+1}} \| f^i_{t_z}, K_i)$ where $T = T_{w+1}$.
14:        insert $f^i_{t_z} = \langle t_z, attr_2, \ldots, attr_p, null \rangle$ in $s_i$_tab
15:        set $last\_tuple = f^i_{t_z}$ and $last\_time = T_{w+2}$
16:    else
17:        search $T_{w+q} : t_z \geq T_{w+q}, q > 1$
18:        update $s_i$_tab.$MAC_i = HMAC(f^i_{t_j} \| d_{T_{w+1}}, K_i)$ where $t = t_j$
19:        and m_tab.$MAC_i = HMAC(d_{T_{w+1}} \| d_{T_{w+q}}, K_i)$ where $T = T_{w+1}$
20:        and m_tab.$MAC_i = HMAC(d_{T_{w+q}} \| f^i_{t_z}, K_i)$ where $T = T_{w+q}$.
21:        insert $f^i_{t_z} = \langle t_z, attr_2, \ldots, attr_p, null \rangle$ where $T_{w+q}$ in $s_i$_tab
22:        set $last\_tuple = f^i_{t_z}$ and $last\_time = T_{w+q+1}$
23:        close
24:    end if
25:    close
26: end if
```

T_{w+q} (with $q > 1$) such that $t_z \geq T_{w+q}$. Then, s_i will perform an update on s_i_tab to change the *MAC* attribute of the previous inserted tuple from *null* to $HMAC(f^i_{t_j} \| d_{T_{w+1}}, K_i)$, two updates on m_tab, the first to set the attribute $MAC_i = HMAC(d_{T_{w+1}} \| d_{T_{w+q}}, K_i)$ for the row corresponding to the *marker* $\hat{d}_{T_{w+1}}$ and the second to set the attribute $MAC_i = HMAC(d_{T_{w+q}} \| f^i_{t_z}, K_i)$ for the row corresponding to the *marker* $\hat{d}_{T_{w+q}}$. Finally, it inserts the new tuple in s_i_tab with the *MAC* attribute set to *null* and also stores it altogether with the time position T_{w+q+1} of the next *marker* in its local memory.

t	*lat*	*long*	snapshot	*MAC*
02:16:57	41.508577	14.238281	b568eiaf38 ...	60177w2455 ...
02:17:02	41.508577	14.238281	b5684wks90 ...	93048ag343 ...
⋮	⋮	⋮	⋮	⋮

Fig. 9.6: An example of s_i_tab.

9.3.2 Database pruning protocol

As stated in Section 9.2, our approach implements also an aging mechanism for automatically deleting older tuples to limit the database size. Deletion is carried out only on discrete time intervals, i.e., only the removal of an entire non-empty bucket at a time is allowed. In this mechanism, the *markers* play a key role. Indeed, as only discrete deletion is allowed, each *marker* represents a milestone maintaining device chains when previous elements are removed. Data owner can always compute the first *marker* in the database still valid.

9.3.3 Integrity verification protocol

Suppose the data owner submits the range query $Q^i(t_l, t_u)$, meaning that all snapshots recorded by device s_i in the time interval $[t_l, t_u]$ should be returned intact as result. Our protocol enforces that the query processor module, cloud-side located, returns the tuples belonging to all the buckets involved in the interval $[t_l, t_u]$ along with all the *markers* linked to elements of such buckets. Moreover, as additional information, the data owner knows the time position of each *marker* and, for each device, knows the last tuple inserted in the database. Observe that, this requirement is easy to obtain because all devices have an internal memory in which this information is stored and we assume that the data owner can access it at every moment. To verify the integrity of the result obtained, the data owner performs the following steps:

1. First, he verifies the head and tail of the chain. Specifically, as for the head he verifies if the time value of the first *marker*, say T_f, is lower than or equal to t_l. Concerning the tail, instead, we can identify two cases: *(i)* t_u is lower than or equal to the time value of the last *marker* of the query result. In this case the tail is verified and no further checks are required. *(ii)* t_u is greater than the time value of the last *marker* of the query result. In this case the owner has also to verify if the last tuple stored by s_i is present in the query result.
2. Then, starting from the first *marker*, he verifies each chain link by iteratively computing the MAC attribute of each element and comparing it with the value returned by the cloud, to verify the query answer integrity.

The steps performed by the data owner to verify a query result are reported in the Algorithm 8.

9.4 Performance Comparison

In this section, we provide a comparison analysis of the performance of our approach w.r.t. Merkle-hash-tree based solutions. The results of this analysis provide a valid support to the motivations that gave rise to our work.

As will be shown in Section 9.6, Merkle-hash-tree based solutions can be considered the state of the art when it comes of deterministic approaches for query integrity. For this comparison, we analyze the cost of two basic operations: (1) insertion of a new tuple and (2) verification of range query results. We also consider the amount of information that must be transferred to verify a query. We measure the computational cost in terms of number of hash computations on a single block on which the compression function operates (which we call *unitary* cost). Let denote by b the size of this block (it is 512 bits for SHA-256). Since the hash

Algorithm 8 Integrity Verification Protocol

```
1: submit the query $Q^i(t_l, t_u)$
2: receive $Ma$: set of $markers$ $Ma = \{d_T : T \in [T_f, T_l]\}$
3: receive $Tu$: a set of tuples
4: read the $last\_tuple_i = \hat{f}^i_{tz}$
5: if $t_l < T_f$ then
6:     return false
7: else if $t_u > T_l$ and $last\_tuple_i \notin Tu$ then
8:     return false
9: end if
10: close
11: for all $t_h \in Tu$ do
12:     if $!checkMAC(t_h, t_{h+1})$ then
13:         return false
14:     end if
15:     close
16: end for
17: close
18: return true
```

function follows the Merkle-DamgåÃěrd scheme, the computation cost of the hash value of a generic message M of size m is approximatively $\frac{m}{b}$ times the unitary cost. Consider that, for plausible numeric values of our tuples, the cost of the hash of a single tuple is lower-bounded by the unitary cost.

As for the insertion, as stated before, our approach guarantees asymptotic constant costs ($O(1)$). Merkle-hash-tree approaches, on the other hand, require a logarithmic computational complexity $O(log\ n)$ in the dimension of the dataset n, as they need to update the tree-based data structure after each insertion. Consider that, in our domain, the asymptotic analysis gives very meaningful information because the dimension of datasets can be huge. Therefore, despite the general efficiency of logarithmic costs, moving from logarithmic to constant costs results in effective benefits. In our application setting, the high frequency of insert operations makes these benefits definitively relevant.

To make a fair comparison, we could think of combining the Merkle-hash-tree approach with a bucket-based strategy. In other words, we could aggregate the lowest level of the tree, by defining k buckets, each of dimension $\frac{n}{k}$ (by assuming uniform distribution), in order to reduce the depth of the tree. Anyway, the same efficiency of our insertion could be reached only for $k = n$ (i.e., for a degenerate tree with just the root). In this case, query verification would require the computation of the hash value of the whole single bucket. Due to the considerations given above, this cost is lower-bounded by n unitary costs.

In contrast, our technique requires $\frac{n}{k}$ unitary hash computations. Observe that, while the aggregated version of Merkle hash tree enforces the reduction of k to obtain efficient insertions, for us the cost of insertion is independent from k, so that we can set k suitably large. Concerning k, we have to observe that besides the number of hash values to compute, also the amount of information that should be transferred to verify a query must be taken into account. Indeed, the smaller k, the higher this amount, as we have to transfer $\frac{n}{k}$ tuples (for this reason, the degenerate case with $k = 1$ has not practical meaning). Therefore, k should be a fortiori as large as possible. By excluding buckets so small that the little overhead produced by the markers becomes non negligible, we are free of increasing the value of k. In principle, we could estimate the distribution of the incoming data stream, and set k in such a way that buckets are equi-depth, thus including a constant number of tuples.

9.5 Security analysis

In this section, we carry out the security analysis of our approach to validate it also in presence of attacks from the cloud server. We describe the security model and analyze the security properties of our proposal. We may identify the following properties our system has to assure:

1. **SP1.** The result of a range query has to be complete, it means that no tuples or attributes are omitted (including *markers*).
2. **SP2.** The result of a range query has to be fresh. It means that the data owner can check whether the newest version of the result is returned.
3. **SP3.** The result of a range query has to be correct. It means that the data owner can verify that all the values in a query result have not been corrupted, nor have spurious records been introduced.

To analyze the security properties above, our threat model includes the following assumptions:

1. **A1.** The camera devices and the data owner are trusted, unbreakable, and dependable.
2. **A2.** A secure communication between database and camera devices is established.
3. **A3.** Collision, preimage and second preimage attacks on the cryptographic hash function, used by $HMAC$, are infeasible.

Now, we are ready to show that our approach satisfies the security properties on the basis of the attack model described above.

Compliance with SP1.

In this case, our approach has to resist attacks in which the cloud server may attempt to return incomplete results. We may identify four possible ways the cloud can reach this goal:

1. The cloud returns an empty set. In this case, the data owner checks whether there exists a last inserted tuple. In the affirmative case, the attack is detected.
2. The cloud tries to omit some tuples of a query result. In this case, the data owner can check that the result is incomplete by verifying the chain linking between all the result elements inside each bucket (see Section 9.3.3). Therefore, thanks to Assumption **A3**, this attack is contrasted.
3. The cloud server does not return a whole internal bucket. Also this attack is detected because, in the first step of the integrity verification protocol, the data owner identifies the chain header and tail and then, in the second step, verifies the whole chain integrity. Therefore, any missing bucket is detected because the corresponding links in the chain will be missing too, thus causing an interruption in the chain.

4. The cloud server does not return elements or buckets at the edges of the query result. This attack makes sense only in the case of *unbounded* query, i.e., for query in which one or both interval ends are not fixed. As for the verification of the left-hand bucket, the owner can always identify the first *marker* because this information is part of the public scheme. This is true also in case of deletion (triggered by the aging mechanism described in Section 9.3.1) because the owner can always compute the first oldest valid *marker*. In the case of a right unbounded query, the owner can verify the completeness of the result by checking whether the last tuple inserted by the considered device is present (see Section 9.3.1).

Compliance with SP2.

To be compliant with **SP2**, our approach should resist attacks in which the cloud server tries to return the query result computed on an old complete version of the database. This kind of attack can always be detected by the data owner because he knows the last tuple inserted in the database. Indeed, the *MAC* attribute of each tuple can be either *null* or assume its final value (i.e., no further updates are allowed after the initialization). Moreover, no *null MAC* attribute value may exist for elements older than the last inserted tuple. Then, an old version of the result can be detected if there exists a tuple older than the last inserted in the database with a *null MAC* attribute.

Compliance with SP3.

The compliance with **SP3** requires that our approach guarantees data correctness. We may identify three cases:

1. The cloud server adds a whole bucket or a single tuple in the query result. However, due to assumptions **A1** and **A3**, no valid *MAC* attribute can be generated without a secret key stored in the device firmware.
2. The cloud server compromises some tuples or *markers* in the query result. Also this attack cannot succeed because of the above reasoning on *MAC* attributes.
3. The cloud server tries to mix data recorded by different devices. As described in Section 9.3.1, each device s_i owns a private key K_i, therefore, due to assumptions **A1** and **A3**, this attack is not possible.

9.6 Related work

Since cloud computing is a successful emerging model the issue of assessing the integrity of the stored data in terms of completeness, correctness and freshness represent a critical and useful aspect to investigate [209].

There exist two types of solutions typically adopted in this context, probabilistic and deterministic solutions. Probabilistic solutions give a not certain (probabilistic) guarantee of detecting violations [244, 82]. Specifically, the proposal described in [244] proposes a probabilistic integrity audit method. Their scheme works through the insertion of a small number of tuples into the outsourced database. For a query issued against this augmented database, there is certain probability that a small amount of the inserted tuples is returned with the original data. An enhancement of this model is presented in [82], which allows to check the integrity of queries (in particular join queries) performed by a non trustworthy server. This approach relies on the insertion of fake tuples in the encrypted data. These tuples are divided in markers (if they are newly generated tuples) or twins (replicas of tuples that already exist). Moreover, tuples are subdivided in buckets by using salts in the encryption, and possibly inserting dummy tuples. The detection of a violation occurs if a twinned record appears solo or a expected marker is missing. Obviously a trade-off exists because the the more the fake tuples inserted, the more the offered guarantee, but in the meantime, the more the fake tuples inserted, the more the overhead.

Deterministic approach is the second family of solutions. One of the simplest data integrity technique, defined in [184], consists in signing each tuple and creating a new attribute in the same data table to store the corresponding signature (i.e., the hash value of the tuple). Once the data owner makes a query, the cloud sends the corresponding hash value along with the tuple data to verify it. Although the signature can prevent data corruption, obviously this scheme cannot assure any completeness requirement. Further solutions, to assure the integrity of data storage, exploit hashing or digital signature schemas [114, 161]. These kind of approaches are deterministic but impose an overhead that is not always bearable in cloud scenarios. There are other possible solutions defined as probabilistic [147, 134] that do not give a certain proof of integrity but use an probabilistic estimation. They leverage some particular data put among the encrypted data [134].

Completeness of range queries results is not only related to the integrity of data storage but also to the correctness of the way the queries are executed. This type of integrity is referred as computation integrity and it has to be guaranteed through the verification of the correctness, the completeness and the freshness of query results.

Also this kind of integrity can be checked through both deterministic and probabilistic solutions. As for the approaches belonging to the first type [160, 246, 120], they can detect integrity violations with certainty usually leveraging some additional data called verification objects, which the server should insert in the response together with the results.

In particular, Pang et. al. [191] use signature chaining schemas and organize tuples in a chain. The chain is structured through an aggregated signature that signs each record with the information from two neighboring records in the ordered sequence. This ensures the result of selection query is continuous by checking the aggregated signature. A similar approach is proposed by Narasimha [179] in DSAC (digital signature chain), where the signature of a record is made up of its immediate predecessor tuple of every searchable dimension and itself. The drawback of the above solutions is that they make a massive use of iterative hash functions and digital signatures: thus, our approach is more efficient.

Moreover, [246, 86, 184] are based on Merkle tree and its variation (i.e., MB-Tree). For instance, the proposal by Devanbu et al in [86] provides authenticity and non-repudiation of the query results exposing to the user the tuples immediately beyond the left and right boundaries of the query, and proving completeness through a verification object. In this case the verification object is a Merkle tree. Some extensions of basic Merkle tree useful to handle the completeness aspect of integrity are investigated in [85]. As shown in Section 9.4, our technique is definitively more suitable than Merkle-tree based approaches to our application context.

A further deterministic scheme, called MAC Chain, is proposed in [120]. This work uses a model based on HMAC to provide a mean to verify query results integrity. Some differences with our solution is that *(i)* the verification process is implemented on the TTP (Trusted Third Party), *(ii)* in order to support query integrity verification, the data owner has to alter the outsourced relation structure. He inserts the verification objects computed by means of a MAC that takes all the tuple values into account. It is clear that in our scenario in which video sensors make a lot of insertions in the database, a low computational cost of insertion is required, hence the model proposed in [120] is not suitable.

10

A Distributed Methodology and its Implementation for Privacy-Preserving Access Accountability in Critical Environments

In this chapter, we present a system to generate privacy-aware logs of people accesses to critical areas. The solve the trade-off between security assurance and user privacy, it implements a k-anonymity approach allowing us to guess who accessed a zone (for example, a room of a building), at a given time, with probability k^{-1}. The value of the parameter k is suitably set in such a way that the uncertainty degree of access logs is enough to guarantee privacy and, at the same time, allows us to restrict suspects to a feasible number of people in case of a security incident. Importantly, privacy is guaranteed against both server and client-side attacks. We present an effective software-hardware solution based on RFID and BeagleBone technologies implementing the above model. A distributed algorithm allows the nodes of the network to independently generate quasi-IDs guaranteeing the above privacy property, and satisfying security requirements. The experimental evaluation shows the capability of the proposed method to reach satisfactory results.

10.1 Problem formulation and solution sketch

In this section, we describe the scenario considered in our study, the addressed problem and we sketch the proposed solution. We consider a surveillant environment (e.g., a museum, a government office, a tribunal, etc.) in which people internal accesses must be logged for security purposes. However, as widely remarked in the introduction, for privacy reasons the tracking must be done introducing a certain degree of uncertainty.

In particular, we require that, given an instant of time τ and a person p, it should be possible to guess the location (with adjustable approximation[1]) of p inside the environment at the time τ with a probability k^{-1}, where k is positive integer number representing a privacy requirement.

Now, we sketch the approach proposed to address this problem. Our technique is based on the use of movement sensors, RFID tags and readers, a server, and a suitable communication infrastructure.

Preliminarily, the environment is partitioned into two types of zones: *monitored* and *non-monitored* zones. The choice of the type of zone depends on the particular scenario: for ex-

[1] We mean that in a real implementation of the proposal, we may decide the degree of approximation accepted for the position of people (e.g., 50 meters), which depends on the partition of the environment, as we will discuss later.

ample, we expect that in an government building, monitored zones are any room, bathroom, and so on, in such a way that in case of necessity it is possible to verify (a-posteriori) the movements (obviously, with uncertainty) of a terrorist, for example to identify possible accomplices.

Moreover, each user to monitor is identified and associated with a passive and cheap RFID tag, which is attached to something that the individual must keep with him: for example, it may be attached to the visitor pass in a government building or to the ticket in a museum.

Each monitored zone is equipped with a movement sensor and a RFID reader to detect the presence of a tag (thus, of a person). Sensors are able to detect the physical movement of a person entering a zone in such a way that in case this movement is not associated with the reading of a RFID tag, an alert is triggered. When a person enters a monitored zone, the reader associated with this zone reads the identifier EPC of the RFID tag. Clearly, data received from readers allow the exact (i.e., with no uncertainty) tracking of people, provided that the association between tag EPC and person identity is known. The risk is concrete because an attacker who accessed such data and knows the association tag-person may track any user.

To avoid this, received EPC are processed to generate new location traces that can be stored guaranteeing users' privacy. In particular, the idea is that the tag EPC is transformed into a new number in such a way that k people are associated with the same number (which we call *qID – quasi identifier*) and such a transformation is one way.

All sensors are connected to a local network so that the so obtained qIDs are sent to a server, which manages a log file.

10.2 System architecture and implementation

In this section, we describe the architecture of our system and provide some details about the technologies and hardware used. Our solution leverages on two mechanisms: the identification of people based on RFID technology and the physical access monitoring.

As for the identification of people, in our design, passive RFID tags are used. They operate at a frequency of 865-868 MHz in Europe and 902-928 MHz in America, have a read range up to 10 meters and a cost of about 0.15$. Each reader can detect hundreds of tags in few seconds and based on the signal strength received, the reader reports or ignores the received EPC to avoid multiple reading of the same tag.

Readers are built by exploiting the BeagleBone Black community-supported development platform: BeagleBone is a single-board computer equipped with open-source hardware particularly suited to be adopted as network device and has two expansion connectors allowing other *capes* (i.e., expansion boards) to be stacked onto it. In particular, the BeagleBone community maintains a project, namely *RFID Adaptor Board* [5], whose purpose is the distribution of a cape supporting the Texas Instrument's TRF7970ATB multi-protocol RFID transreceiver (Figure 10.1). This particular cape allows us to built our RFID readers by using BeagleBone Black boards equipped with the TRF7970ATB RFID module. BeagleBones used in our sys-

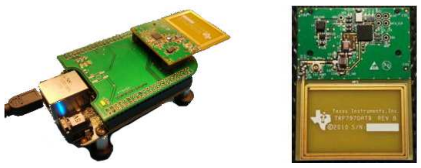


Fig. 10.1: The BeagleBone with the RFID Adaptor Board and the Texas Instrument's TRF7970ATB RFID module

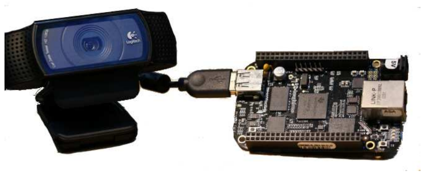


Fig. 10.2: The BeagleBone and the Logitech C920 webcam

tem run a software implementing the data processing logic that allows us to obtain privacy preserving logs (this issue is discussed in Section 10.3).

As for the physical access monitoring task, which is the second mechanism on which our solution leverages, it can have several implementations and the choice is strictly related to the particular environment under analysis and the precision requirement. The most common solutions are the installation of *turnstiles* allowing people to pass through them only if a tag is detected, *infrared beams* and *thermal people counters* or *video counting* which are considered as enhanced people counter solutions.

Concerning the solution based on turnstiles, this is the most invasive one and can only be adopted in specific and limited environments where heavy structural modifications can be carried out. Moreover, it is not possible to verify that a tag belongs to the person crossing the turnstile. Infrared beams and thermal people counters can be easily installed but they still do not allow the verification of the tag and the person crossing the sensors. The last solution is the most advanced and complete one, however the use of computer vision techniques could lead to some privacy concerns if not properly handled.

In our design, we adopt video counting to stick with the most enhanced and powerful solution. To reduce privacy issues related to handling of face images, our solution operates on the fly not storing any image. The image processing software has been developed by using the OpenCV library [4] and installed on the BeagleBone Black devices equipped with the high resolution cam Logitech C920 (Figure 10.2). Actually, both the RFID people identification and the physical access monitoring modules are installed on the same BeogleBone Black board.

The deployment diagram of our system is illustrated in Figure 10.3. The BeagleBone executes two modules, namely the Video Processing and the RFID Engine. The former communicates with the Logitech C920 and executes the `imgRecognition.jar` java software to perform the physical access monitoring task. The latter, instead, receives information about RFID tags in the proximity of the RFID reader stacked on the board and performs our strategy for privacy-preserving log storing by means of the java software `idTrasformation.jar`, which is described in the next section.

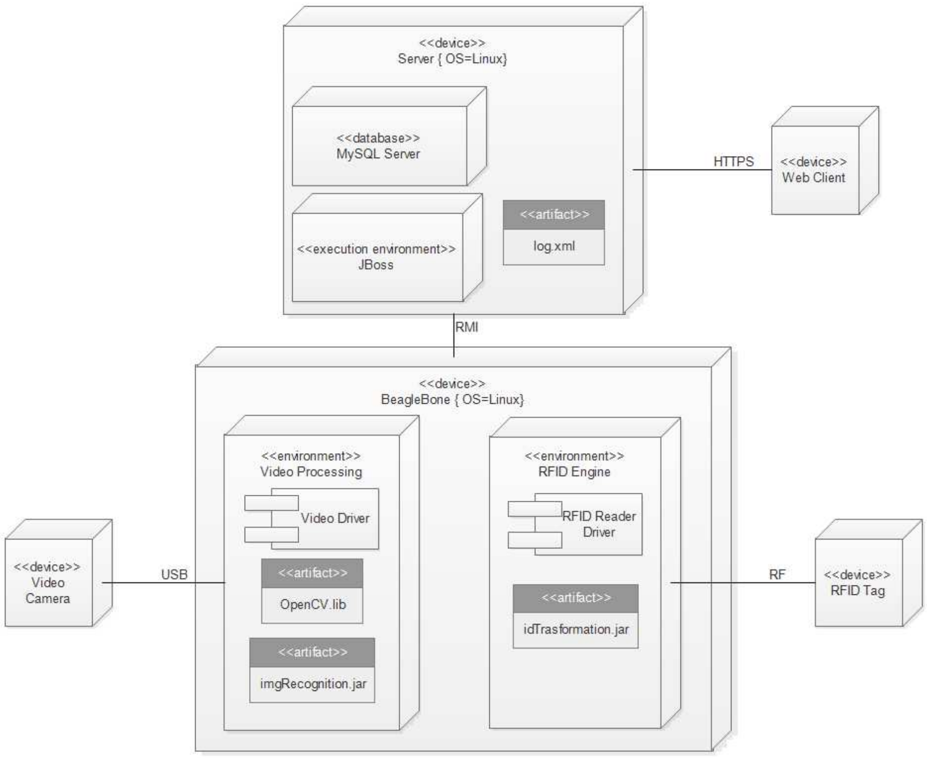


Fig. 10.3: The deployment diagram of our system.

The BeagleBone boards, installed at the entrance of each zone that has to be monitored, communicate with the central server, which is in charge of storing the anonymized information about user access providing remote access to the log. An example of the connections among BeagleBones and server are sketched in Figure 10.4.

10.3 qID generation

The notion of *k-anonymity*, introduced above, means that the probability of identifying a person by accessing the stored location traces is k^{-1}, where k is a given anonymity requirement. Our system reaches this goal by suitably transforming the received EPC into a new number called qID (i.e., quasi identifier) belonging to a new domain. This domain is smaller than the number of people so that more people are associated with the same qID. In particular, this domain is the interval $[1, d]$, where d is a positive integer system parameter.

The log file stores four types of events, which are described in the following.

1. *Entering the building*. When a person enters a government building, one RFID tag is given to (and associated with) him. This tag is integrated into the *document* issued to permit the access to the building. Observe that, RFID tags used in our proposal have a life cycle.

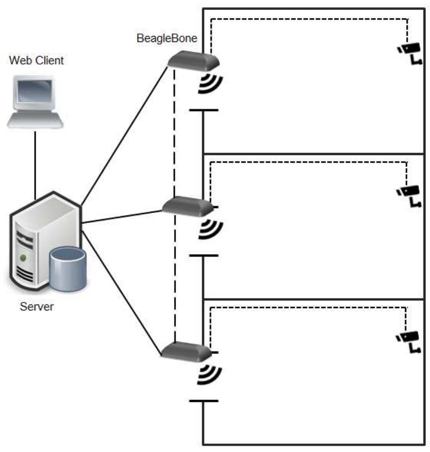


Fig. 10.4: A sketch of our system architecture.

Indeed, in a real scenario, there must be a large number of RFID tags *ready* to be given to people. After a tag is associated with a person, it starts to work and we say it becomes *working* (active).

Each BeagleBone needs to store the EPC of the currently *working* tags in the sorted set S. When a tag becomes working, a message containing the timestamp, the type of event (activation of a tag), and the assigned EPC is sent through the network. This way, each BeagleBone can add into S the new EPC.

2. *Leaving the building*. When a person leaves the building and returns the tag, this tag ends its life becoming *inactive*. Also in this case, a message containing the timestamp, the type of event (deactivation of a tag), and the tag EPC is sent through the network. All BeagleBones listening this event remove from S the EPC of the inactive tag.

   Note that, tags used in our proposal are not *one-time*, because they will be reused for another person after they are deactivated. As it will be clear in the following, this does not rise any problem from the security point of view.
3. *Entering a zone*. When a person enters a controlled zone, the BeagleBone associated with this zone reads the *EPC* of his tag, which is typically a 96-bit identifier. A new privacy-preserving qID associated with this tag is computed from the EPC by means of the following three transformations:
   a) The first transformation maps the 96-bit domain of EPCs, which is highly *sparse* (because the number of tags used is much smaller than 2^{96}) into the dense domain $[1, |S|]$. Specifically, the received EPC is mapped to the integer p, which is the position of the EPC in the sorted set S, with $1 \leq p \leq |S|$.
   b) The second transformation is implemented by a *random permutation function* RPF having domain and codomain equal to integer interval $[1, |S|]$, which is used to suitably exchange the position of two integers in this interval. Specifically, this transformation is implemented by the function $RPF : \mathbb{Z}_t^* \rightarrow \mathbb{Z}_t^*$ where $\mathbb{Z}_t^*$ is the multiplicative group of $\mathbb{Z}_t$, $\mathbb{Z}_t$ is the set of (equivalent classes) of integers (**mod** t), and

EPC	119	182	190	200	310	480
1st transf.	1	2	3	4	5	6
2nd transf.	4	1	5	6	3	2
3rd transf.	2	2	3	1	1	3

Table 10.1: An example of the transformation of 6 EPCs.

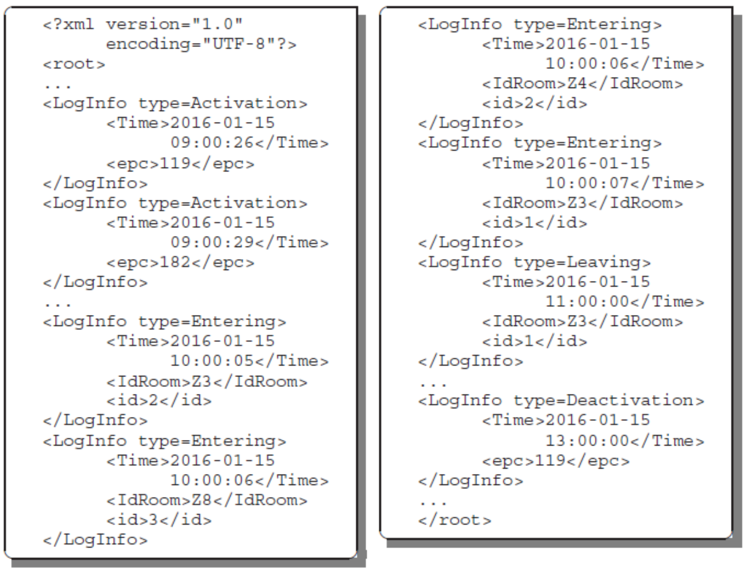
$RPF(i) = i \cdot g \ (\textbf{mod}\ t)$, for any $g \in \{1,\dots,t\}$. RPF works as a permutation, because $\mathbb{Z}_t^*$ is an additive cyclic group, and every $i \in \{1,\dots,t\}$ is a generator for $\mathbb{Z}_t^*$ when t is prime. After this step, the position p is transformed into the *permuted* position $p' = RPF(p)$.

c) In the last transformation, a hash function H having domain $[1,t]$ and codomain $[1,d]$ is adopted (we recall that d is a system parameter), where $d \leq t$. This function has the purpose of reducing the domain of the output (i.e., the stored qIDs) and generating the collisions among more qIDs necessary to implement the k-anonymity approach. Also in this case, several implementations of this function can be used: we tested our system by adopting the simplest hash function $qID = 1 + p' \ mod \ d$.

To help the reader understand the whole qID generation process, we provide in Table 10.1 the instantiation of the above steps done in our system with parameter $d = 3$. We assume that the sorted set of active tags is $S = 119, 182, 190, 200, 310, 480$, where, for the sake of presentation, we replace the 96-bit string of EPC by a 3-digit decimal number. The first row reports the EPCs of the tags of the involved people. The second row shows the result of the first transformation, which is the position of each EPC in the set S. The third row reports the result obtained by applying the random permutation function, and, the last row shows the use of our hash function to generate the final qIDs.
When a user accesses a monitored zone, the procedure described before is triggered and a new qID is generated for him. After the computation of a qID, a message containing the timestamp, the type of event (entrance into a monitored zone), the reference to the zone entered and this qID is sent through the network. Moreover, the BeagleBone involved in the reading adds into a local map T the pair ⟨EPC, qID⟩. This information is used when the tag will leave the zone but is available only to the reader (i.e., no method is implemented into the reader firmware to access the map T).

4. *Leaving a monitored zone*. The last event that is logged occurs when a person having a tag with a given EPC exits a monitored zone. In this case, the sensor searches for the read EPC into the map T to obtain the qID previously associated with this EPC. If such a query does not give any result, an alarm is generated to report the fault situation. Otherwise, a message containing the timestamp, the type of event (leaving from a monitored zone), the reference to the zone and this qID is sent through the network. Moreover, the pair ⟨EPC, qID⟩ is deleted from T.

All the messages sent through the network are appended to the log file stored by a server. The log is encoded by XML format and does not include the information necessary to associate a person with a tag, which is expected to be stored elsewhere. Each log event is mapped into an



```
<?xml version="1.0"
      encoding="UTF-8"?>
<root>
...
<LogInfo type=Activation>
      <Time>2016-01-15
            09:00:26</Time>
      <epc>119</epc>
</LogInfo>
<LogInfo type=Activation>
      <Time>2016-01-15
            09:00:29</Time>
      <epc>182</epc>
</LogInfo>
...
<LogInfo type=Entering>
      <Time>2016-01-15
            10:00:05</Time>
      <IdRoom>Z3</IdRoom>
      <id>2</id>
</LogInfo>
<LogInfo type=Entering>
      <Time>2016-01-15
            10:00:06</Time>
      <IdRoom>Z8</IdRoom>
      <id>3</id>
</LogInfo>
```

```
<LogInfo type=Entering>
      <Time>2016-01-15
            10:00:06</Time>
      <IdRoom>Z4</IdRoom>
      <id>2</id>
</LogInfo>
<LogInfo type=Entering>
      <Time>2016-01-15
            10:00:07</Time>
      <IdRoom>Z3</IdRoom>
      <id>1</id>
</LogInfo>
<LogInfo type=Leaving>
      <Time>2016-01-15
            11:00:00</Time>
      <IdRoom>Z3</IdRoom>
      <id>1</id>
</LogInfo>
...
<LogInfo type=Deactivation>
      <Time>2016-01-15
            13:00:00</Time>
      <epc>119</epc>
</LogInfo>
...
</root>
```

Fig. 10.5: A partial representation of a log.

element `LogInfo`. The attribute `type` specifies the typology of event, which can be activation or deactivation of a tag, or the reading of a tag when a person enters or leaves a zone. `LogInfo` sub-elements are: the timestamp of the event, the read EPC, the identifier of the involved zone and the generated qID.

An example of this file, concerning events occurred in the morning of 20 July 2015, is reported in Figure 10.5. The first two `LogInfo` elements show that two people accessed the building at about 9 and the tags with EPC 119 and 182 (again, for simplicity, we represent an EPC by 3 digits) were assigned to them. The next fragment of the log reports four entries into the zones Z3, Z8, Z4 and again Z3, occurred at about 10. Observe that the last information is the qID generated according to the procedure described at Step 3. Then, the log registered the leaving from zone Z3 of a person associated with the qID=1. Finally, the last `LogInfo` element shows that the person, to whom the tag with EPC=119 has been assigned, has left the building at 13:00.

10.4 Log Analysis

In case of necessity, it is possible to elaborate the log file to find the set of the users who could enter a given zone at a given time T. We say "could enter" because this operation returns a set of k EPCs, in order to comply with the privacy requirement. For example, consider the third `LogInfo` item of the log file fragment reported in Figure 10.5, we could be interested in knowing (with uncertainty) who is the person who entered with qID=2 the zone Z3 at T =10:00:05.

Algorithm 9 shows how to solve the above problem. It receives three inputs. The first one is the index n of the log line to analyze. The other two inputs (i.e., T and S) are optional and have as default value $T = 0$ and $S = \{\}$ (their use will be clarified later). To solve the problem, the algorithm computes the set S of active tags at time T (Lines 1-11). This is done by elaborating log files from the beginning, and adding to (Line 6) or removing from (Line 8) S an EPC each time an event of tag activation or deactivation is found in the log. To accelerate this computation, it is possible to create periodically a sort of *check point*, i.e., to save the set S' of the active tags at a given timestamp T'. In this case, giving as optional input also S' and T', then the algorithm skips all log lines related to the events occurred before T' (Line 3). As a consequence, it is possible to start the construction of the set of active tags from this point instead of the beginning of the file.

After the set S has been obtained and the n-th line has been reached, a check that this line concerns an entering event is done (Line 12). Then, the inversion of the 3^{rd} (Line 13) and 2^{nd} (Line 15) transformation (see Section 10.3) are done to obtain the position p of the considered EPC in S. Finally, this EPC is added to the output.

We provide now an example of this computation. Suppose we want to analyze the third `LogInfo` item of the log file fragment reported in Figure 10.5 to know (with uncertainty) who is the person who entered the zone Z3 with qID=2. Assume that the active tags are those reported in the first row of Table 10.1. The first operation to do is to find the numbers $x \in [1,6]$ such that $1 + x \ mod \ d = 2$. As $d = 3$, we have two candidates, $x_1 = 4$ and $x_2 = 1$. Then, by inverting the random permutation function, we find that x_1 and x_2 derives from 1 and 2. As a consequence, the requested EPC are the first and the second tags of S, which are 119 and 182. Thus, by restoring information from the third `LogInfo` item of the log fragment shown in Figure 10.5 with the above procedure, we gather that at 10:00:05, the person associated with the EPC equal to 119 or to 182 was in the zone Z3.

Finally, observe that by an analogous procedure is possible to analyze also items of the log relative to events of leaving type.

10.5 Validation

In this section, we describe some experiments carried out to show how our system behaves and to test its performances.

We remark that experiments are aimed at validating our approach and no comparative experiments can be done. Indeed, as highlighted in Section 10.6, no privacy-preserving localization method exists in literature able to resist server-side attacks.

We simulate the application of our approach to a real italian government building, namely the Ariano Irpino law court[2]. It consists of 2 floors and 56 rooms (zones) and a representation of its planimetry is reported in Figure 10.6. Concerning the setting of RFID readers, we assume that there is a reader at the entrance of each monitored room of the building and that the adjustable range of the RFID readers is set to detect tags which are at a maximum distance of 50 cm (about half the size of a door) from the reader. Since, for security reasons, all people (employees or visitors) inside the building must already bring a badge with them, RFID tags can be embedded in such badges. Each time a person crosses the entrance of a

[2] http://www.tribunalearianoirpino.it

Algorithm 9 *EPSRestoring*

```
Constant  $d$: system parameter
Input  $n$: the line index of the log file to analyze
Input  $T$: a timestamp
Input  $S$: the set of active tags at timestamp $T$
Output  $U$: the set of candidates
Variable  $X$: a set of integers
Variable  $p$: an integer
 1: for $n$ times do
 2:     read next line $L$ from log
 3:     if $L$.timestamp > $T$ then
 4:         if $L$.type = activation then
 5:             $S$.add($L$.EPC)
 6:         end if
 7:         if $L$.type = deactivation then
 8:             $S$.remove($L$.EPC)
 9:         end if
10:     end if
11: end for
12: assert ($L$.type = entering)
13: $X = \{x \mid (1 + x \ mod \ d = L.\text{qID} \}$
14: for each $x \in X$ do
15:     $p = RPF^{-1}(x)$
16:     $U$.add($S[p]$)
17: end for
```

room, a reader reads the EPC from the RFID tag and triggers the execution of our technique for creating privacy preserving logs. To simulate people movements inside the building, we built a java prototype implementing a mobility model based on the Random Waypoint Model [29], which has been largely investigated in the literature [171, 104]. Specifically, the mobility model operates as follows: at the beginning of the simulation, each person randomly selects a destination room inside the building. Then, he moves towards it with a velocity selected uniformly at random in the interval (S_{min}, S_{max}). Once a patient reaches the end-point, he stops for a time period that varies in the interval (P_{min}, P_{max}). After this interval, he continues by choosing another room and starts moving towards it. In our simulation, we set $S_{min} = 0.1$ m/s, $S_{max} = 2.1$ m/s, $P_{min} = 0.0$ seconds and $P_{max} = 1$ hour.

10.5.1 Setting the system parameter

The first experiment aims at discussing about how to set the value of the parameter d in such a way to obtain the desired privacy requirement. We recall that the parameter d represents the cardinality of the set of qIDs that can be generated. We want that, at each time, the set of the people with a given qID has cardinality equal to k. In this way, the probability of identifying any individual from the transformed/permuted EPC is k^{-1}, if no additional information is provided.

Observe that a solution always exists for any plausible k: indeed, the trivial solution $d = 1$ always solves the problem because all people are associated with the same qID. Clearly, this

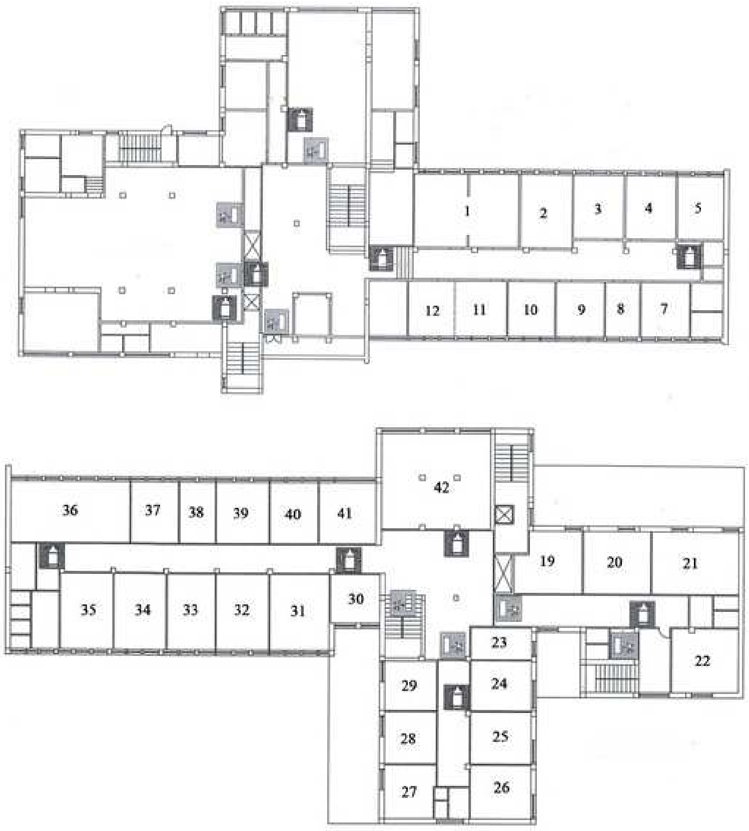


Fig. 10.6: The planimetry of the environment considered.

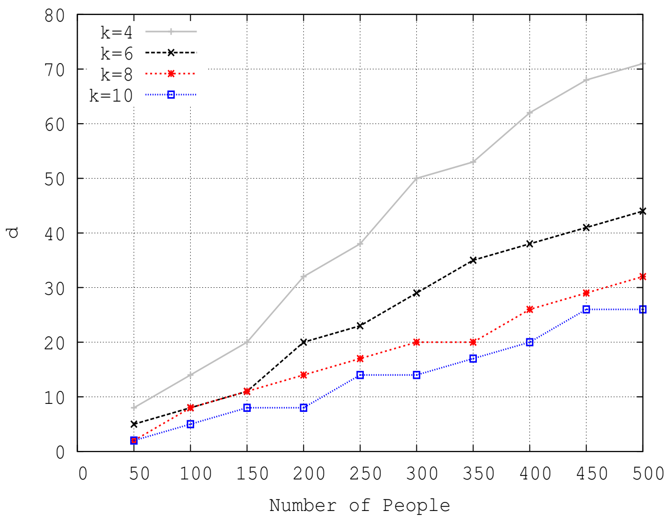


Fig. 10.7: Greatest value of d guaranteeing k-anonymity.

solution has the drawback that no meaningful information is provided. As a consequence, we should find the greatest d satisfying the above property.

In this experiment, we vary the number of people from 50 to 500 and measure the greatest value of d guaranteeing k-anonymity for different values of k. The results of this experiment are reported in Fig. 10.7.

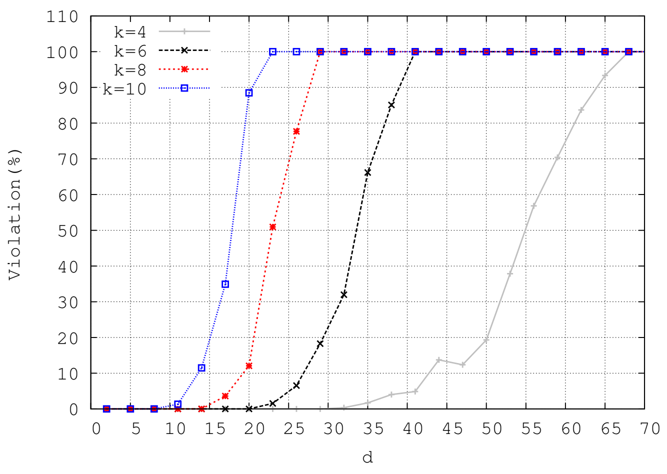


Fig. 10.8: Violations versus size of qID domain.

We observe that all the curves have similar growing trend, because the number of collisions increases as the number of people increases, so that a higher value of d can be selected. The oscillations from the exactly-linear trend are due to the non-ideal behavior of the hash functions and the collisions of people with the same qID in the same room. The analysis of this figure allows us to affirm that our approach is able to guarantee people privacy for any plausible anonymity requirement.

10.5.2 Impact of d

Through the experiment described in this section, we want to measure the impact of the size of the domain of possible qIDs on the capability of our approach of guaranteeing the privacy requirements. For this purpose, we consider the same four values of the parameter k as done in the previous experiment, we set the number of people in the building to 200 and we measure the number of times, called $Violation$, in which the anonymity requirement has not been satisfied. Observe that, this metrics takes also into account the situation in which two users with the same qID are in the same room. Indeed, this event does not generate uncertainty and, therefore, reduces the chances to satisfy the anonymity requirement.

From the analysis of these results, we can state that the parameter d has a strong impact on the performance of our system. Indeed, small variation of this parameter may result in high performance variations. For instance, consider the curve $k = 10$. From the results of the previous experiment we have that $d = 8$ is the value guaranteeing the privacy requirement when the number of people is 200. By means of this experiment, we find that by varying d from 8 to 22 the level of privacy given by the system quickly decreases and reach the lowest value, as people are uniquely identified in all cases.

10.5.3 Impact of the number of monitored zones

In this experiment, we study the effect of the percentage of monitored zones w.r.t. the total number of monitored zones in the building. To this extent, we fix the number of people in the

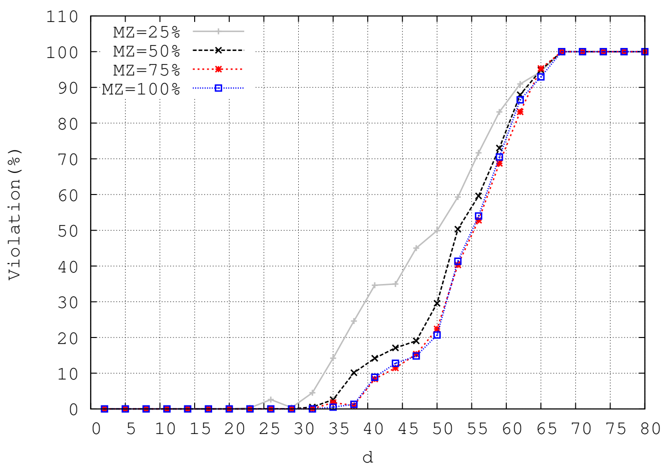


Fig. 10.9: Violations versus number of critical zones.

building to 200 and the anonymity requirement $k = 4$. Now, we vary the percentage of monitored zones MZ from 25% to 100% and we measure the same metrics $Violation$, introduced in the previous experiment, against different values of the parameter d. Figure 10.9 shows the obtained results.

The analysis of this figure allows us to conclude that the number of monitored zones slightly influences the capability of our approach to guarantee the anonymity requirement. Indeed, only when MZ assumes the lowest value (i.e., 25%) it is necessary to reduce the optimal value $d = 35$, as estimated in the previous experiments, to $d = 28$. In all the other cases, the variation of the number of monitored zones has negligible impact. In conclusion, this experiment shows that in the setting of the system parameters it is sufficient to consider only the geometry of the building and not the number of monitored zones.

10.5.4 Impact of time

So far, we have tested the capability of our system to guarantee the privacy requirements by analyzing a single log item. Now we provide a further study aiming at measuring the uncertainty in the identification of a person who entered a zone over time. As a matter of fact, at each access, k possible candidates are determined. Clearly, when more people enter a zone over time, the number of candidates grows. In this experiment, we focus on a given zone and measure the number of candidates (as fraction of the total number of people inside the building) who access it over time. Clearly, the result of this experiment is strongly influenced by the access frequency of this zone. Therefore, we consider four typologies of zone, each characterized by a different *bias* of access *probability* (PB, for short). In particular, when $PB = 0$ we mean that all building zones have the same probability of being accessed, whereas $PB > 0$ means that the probability of accessing this zone is increased by PB w.r.t. the uniform probability. The analysis is conducted by considering a time window representing a whole morning (i.e., from 9.00 AM to 01.00 PM), and fixing the number of people in the building to 200 and $d = 35$. The results of this experiment are reported in Figure 10.10

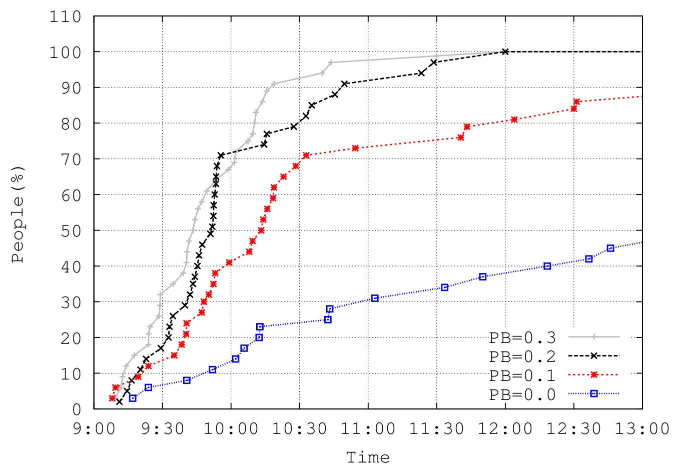


Fig. 10.10: Percentage of people versus time for different values of *PB*.

We observe that, at the end of morning the set of people who could access a zone with no *bias* (i.e., $PB = 0$) is about 47%, whereas when the zone access frequency is higher (i.e., $PB = 0.3$), the set of candidates coincides with the whole population at about 12.00*am*. As a consequence, given an acceptable uncertainty level, this experiment allows us to estimate at which time this level of uncertainty is reached, thus allowing to perform suitable counter-measures. For example, to limit to 15% the possible suspects of a malicious action in a room of a museum with $PB = 0$, we should schedule an inspection per hour to detect *damages*.

10.6 Related work

In this section, we survey the literature regarding location anonymity.

The concept of k-anonymity was originally designed by Samarati and Sweeney in the field of database privacy. According to them, a database provides k-anonymity if explicit identifiers of all the tuples are removed from the database and, additionally, the quasi-identifiers (set of attributes leaking confidential information) of each individual in the database cannot be distinguished from those of at least $k-1$ other individuals. Hence, this approach of k-anonymity suggests the suppression and generalization (obfuscation) of quasi-identifiers.

The general idea of k-anonymity location [107, 185] is that the position of a user is given provided that the probability of identifying her is less than k^{-1}. In many cases, a trusted third-party is necessary to implement such a solution [135]. However, the efficacy of these approaches strongly rely on user density. Indeed, a large number of users must subscribe to the service, otherwise either the privacy cannot be guaranteed or the user's location has to be given with a very large approximation. Moreover, the presence of malicious users strongly compromise privacy. A similar approach to protect location data consists in creating areas of confusion where the traces from several users converge [189]. Also in this case, whenever low-density areas are considered, a threshold between privacy and data accuracy has to be chosen.

Over time a wide range of application scenarios took advantage of this technique to guarantee privacy. This is the case of LBSs and of [24], in which the authors propose an approach for adaptive location-privacy protection in participatory sensing context. In the context of LBSs (location based services), k-anonymity location requires that location information inside a message sent from a mobile user to a LBS should be indistinguishable from at least $k-1$ other messages from different mobile nodes [112]. In this particular application scenario, deleting identifiers from location data is not a good choice. Indeed, as found in [146, 119], several techniques can be exploited to infer the location of a subject's home and, then, the subject's identity. The most popular solution for designing privacy-preserving LBSs consists in obfuscating the actual location from which a query is made by constructing cloaking regions that contain the locations of k anonymous users.

This technique, called location cloaking approach [127, 140], aims to perturb location data by introducing random noise in order to guarantee user's privacy. However, due to the spacial nature of data, in many cases, the original data can be closely estimated from the perturbed data using a spectral filter that exploits some theoretical properties of random matrices [136].

Another privacy-preserving mechanisms is presented in [75]. In this paper, the authors argue that the ability of generating fake contextual data can be important also in privacy preserving applications. For instance, it could be possible to add some dummy queries (or fake transactions) that are indistinguishable from the real queries, so that it is different to trace the real interests of users performing them.

An extension of k-anonymity for specific types of data, like spatio-temporal data, is the (k,δ)-anonymity [23], which is specifically designed for uncertain trajectories defined as the movement of an object on the surface of the Earth. This technique exploits the spatial uncertainty $\delta \geq 0$ in the trajectory recording process. In [230], the authors prove that, for any $\delta > 0$ (that is, whenever there is actual uncertainty), (k,δ)-anonymity does not hide an original trajectory in a set of k indistinguishable anonymized trajectories.

The wide diffusion of RFID technology and the assumption that, according to RFID standard, each object in the network is uniquely identifiable, give raise to privacy concerns. For example, an adversary might be able to discover person's movements, and gain an advantage. Several techniques including privacy-preserving RFID technologies [34], such as EPC re-encryption and killing tags [133], have been proposed to address the privacy issues in the data collection phase where communication between tags and readers takes place.

11

A Privacy-Preserving Localization Service for Assisted Living Facilities

This chapter aims at describing a novel localization service to monitor the position of residents in Assisted Living Facilities (ALF). The service supports a configurable balancing between precision and privacy, in such a way that the right of the residents to move freely in the environment in which they live without being tracked is preserved. However, in case of need, they can always be quickly localized. To do this, we implement, on top of an RFID-based architecture, a probabilistic model guaranteeing that the probability of identifying a person in a given (sensitive) place is k^{-1}, where k represents the required privacy level. This is obtained by ensuring that the EPC sent by RFID tags is not an identifier, but is equal to that of other $k-1$ people, each afferent to a different reader. We show that our method reaches the goal, resisting also attacks aimed at breaking privacy on the basis of humans' movement models. Importantly, privacy is guaranteed against both misuse of the administrator and client-side eavesdropping attacks.

11.0.1 Motivation

The aim of this section is to highlight the importance of privacy in the context of ALFs, together with the relevance of the ALF scenario as application setting for innovative IT solutions. This represents a strong motivation of our work, in which we try to investigate both the above aspects in the context of people localization.

As a matter of fact, ALFs are a good example in which a challenging problem is to find the correct balancing between high services and high privacy. Since many years, researchers are facing this problem. See for example [117], in which the project team considered privacy an important aspect of the environment, as prior research found that residents had strong preferences for privacy. Still in this context, [116] classifies ALFs on the basis of the privacy level required by their patients. The results show that an increasing number of ALFs consider patient privacy as a fundamental feature and, therefore, it is included as one of the main aspect in the evaluation and classification of such environments. Moreover, the Assisted Living Facilities Association of America [26] suggests a general interpretation of assisting living to include a philosophy that emphasizes some form of resident independence, autonomy, and privacy, thus recognizing the importance of residents' privacy as related to the dignity of individuals. The impact of privacy requirements and needs on the organization of ALFs is also highlighted in the book "Assisted living: Needs, practices, and policies in residential care for the elderly" by Zimmerman [254]. In this work, the author discusses the case in which the main reason for nursing home residence is not related to health care. In this context, he clinches the importance of including, in the social model of care, humanistic concerns for high order needs among which privacy is one of the most prominent.

Following the observation above, our approach applied to this application scenario has, therefore, a considerable relevance. The importance of such a scenario is also witnessed by the attention toward it reserved by a lot of other researches in the IT field. Consider, for instance, the survey on IT tools to support ALFs described in [198], in which the authors report that the use of mobile and wearable sensors are becoming pervasive in such environments to improve patient security. Such technologies allows for the implementation of HAR (Human Activity Recognition) modules of fundamental importance in taking care of elderly residents. Another approach showing the importance of the use of IT tools to improve the assistance offered in ALFs is presented in [148]. Here, the benefit of using concepts for home automation environments to take decisions on the basis of variation in the value reported by sensors is proved also by focusing on privacy issues in the storage of patient's data.

The high number of research efforts towards the improvement of services offered in ALFs together with the lack of refined solutions protecting the privacy of people living in these environments are the premises of this work, whose aim is to add a missing piece in the related scientific literature by proposing a solution to guarantee residents' privacy in ALFs still satisfying monitoring requirements.

11.1 Privacy-preserving localization

In this section, we describe the scenario considered in our study and the proposed solution. We consider an assisted living facility in which the position of residents has to be monitored. The approach we follow to address this problem is based on the use of active RFID tags, RFID readers, and a server.

Preliminarily, we introduce the notations used in the following (they are summarized in Table 11.1). Our solution is parametric w.r.t. two numbers d and a: d is a positive integer and a is a real number in the interval $[0,1]$ (their role will be discussed in Section 11.3.2). Let p be the number of residents to monitor and r be the number of RFID readers present in the environment. *RPF* is a *random permutation function* operating on the set of integers $[1,p]$, defined as:

$$RPF : \mathbb{Z}_p^* \rightarrow \mathbb{Z}_p^*$$

where $\mathbb{Z}_p^*$ is the multiplicative group of $\mathbb{Z}_p$, $\mathbb{Z}_p$ is the set of (equivalent classes) of integers ($\mathbf{mod}\ p$), and $RPF(i) = i \cdot g\ (\mathbf{mod}\ p)$, for any $g \in \{1,\ldots,p\}$. We assume that p is prime: this way, *RPF* works as a permutation because $\mathbb{Z}_p^*$ is an additive cyclic group, and every $i \in \{1,\ldots,p\}$ is a generator for $\mathbb{Z}_p^*$.

We are ready to present how our proposal works. We start from the infrastructure. The first operation is to partition the ALF into *cells* by properly positioning the RFID readers in

a,d	system parameters
p	number of residents
r	number of readers
RPF	random permutation function

Table 11.1: Notations

known locations. The position of each reader depends on the power transmission level and the read distance and has to guarantee that each portion of the place is covered by at least one reader.

As for the spatial distribution of readers, we used the approaches proposed in [231]. This solves the problem arising when a tag is located within the range of two RFID readers, which might be falsely identified as to be in two different places at the same time. Each resident is equipped with an active RFID tag that replies to the reader requests sending a suitable number said QID (quasi-identifier). Finally, the readers communicate with a server, which handles data flow on the network.

The idea is that the QID generated by each RFID tag changes at each reading in a pseudo-random way. As a consequence, a malicious attempt to track a resident by sniffing the generated number fails, if the attacker cannot associate the generated QID with the victim. To protect user privacy, we remove any one-to-one correspondence between QIDs and residents, which is replaced by a one-to-many correspondence. However, to preserve the localization service, we require that the above correspondence guarantees the localization of a resident with a probability k^{-1}. In other words, our approach implements a mechanism able to guarantee, at any time, that there exist k different positions in which a resident could be.

The workflow of our protocol to monitor residents is iterative and, in the following, we describe what happens at the iteration $t > 0$.

1. *Query*. The server sends a *reading* request to all the r readers, which, in turn, query the tags inside their coverage area.
2. *QID generation*. Each RFID tag contains a 64-bit pseudo-random number generator (PRNG), whose seed is known by the server, and is able to compute the random permutation function RPF defined above. To reply the request, the tag computes $ID(t) = RPF\big(ID(t-1)\big)$, where we assume that $ID(0) = i$ for the tag of the i-th resident. In words, at the beginning ($t = 0$), each resident is identified by a progressive integer $1, \ldots, p$ and, after each iteration, it is associated with a new integer in $[1,p]$. The random permutation function assures that, at each iteration, two different tags are not associated with the same integer. Then, the tag computes QID at the iteration t as:

$$QID(t) = \begin{cases} ID(t) \bmod d & \text{\textit{if} } PRNG(t) \leq a \cdot 2^{64} \\ \\ PRNG(t) \bmod d & \text{\textit{otherwise}} \end{cases}$$

where, we recall, a and d are system parameters and $PRNG(t)$ is the t-th element of the pseudo-random sequence of the tag. In words, with probability a, the tag returns an hash obtained by uniformly mapping $ID(t)$ to a domain of d elements, whereas, with probability $(1-a)$, $\widetilde{ID}(t)$ is pseudo-randomly generated.

3. *Response*. The j-th reader (with $1 \leq j \leq r$) processes the received data, say $\{QID_1, \ldots, QID_z\}$. For each received QID_i, the tuple $\langle QID_i, x\rangle$ is sent to the server, where x is a positive integer obtained as:

$$x = \begin{cases} R\Big(\{1,\ldots,r\}/j\Big) \; if \, \exists \; z < i : QID_z = QID_i \\ \\ \qquad j \qquad\quad otherwise \end{cases}$$

   In words, if more tags generate the same QID, only the first is associated with j: the remaining ones are associated with other randomly chosen readers (the function R generates a random integer in [1,p], j excluded). Otherwise, if a QID is generated by only one tag, then this tag is associated with this reader (i.e., j).
4. *Storing*. The server collects the tuple received from all readers. If the number of the received tuples is lower than the number of residents, the service detects that a resident has left the ALF and throws an alert. Otherwise, these tuples are stored overwriting the tuples at the previous iteration.

The protocol described above gives us the possibility to locate a resident and to verify that no resident leaves the ALF.

The procedure to locate a resident u is as follows. Let assume we are at the t-th iteration. The server can obtain the $QID(t)$ generated by u, as it can compute $PRNG(t)$ and $ID(t)$ of step *QID generation*. Then, the server performs a *reading* request to all the r readers and among all the tuples received, it filters out the set $T = \{\langle QID(t), l_1\rangle, \ldots, \langle QID(t), l_{|T|}\rangle\}$ (i.e., those referring to $QID(t)$). Now, the server can guess the location of u with probability $|T|^{-1}$: indeed, the possible $|T|$ locations of u are inside the coverage areas of the readers $l_1, \ldots, l_{|T|}$.

Concerning the procedure to guarantee that residents are confined inside the ALF, it works as follows. Periodically, the server executes a *reading* request to the readers and checks that all the p QIDs from the residents' tags are received. When the number of received QIDs is less than the expected one, an alert is generated. Moreover, from the knowledge of the expected QID, it is possible to guess (with a given probability) who is the absent resident and his/her last possible (k) positions (indeed, the server stores the last tuples received). Observe that the frequency at which tags send the signal represents an important privacy issue because a high sending frequency could allow an attacker to track residents' movements. We study this aspect in Section 11.3.2.

11.1.1 Running Example

In this section, we sketch a simple example to show how the whole protocol works and the messages exchange among the actors.

System parameter	Value
a	1
d	2
p	4
p'	5
r	100
t	1

Table 11.2: System parameter settings for the running example

As for the parameter settings, we choose $a = 1$, so that $QID(t)$ will be deterministically generated (see Section 11.1). Moreover, we set the number of possible different $QIDs$ $d = 2$. The number of monitored persons p is set equal to 4 and, therefore, $p' = 5$ (i.e., the first prime number such that $p' > p$), whereas the number of readers is equal to 100. For the sake of simplicity, we omit the discussion about how the system parameters are set on the basis of the privacy requirements. In Table 11.2, we report a summary of the settings chosen for our parameters.

In Fig. 11.1, we illustrate the messages exchange among the actors from the initial state ($t = 0$) to the first iteration ($t = 1$). In the first step performed by our protocol, the server sends the message *reading request* to all the 100 readers (step *Query* of Section 11.1). This starts the iteration $t = 1$, which we are considering in this example. All the readers carry out authentication with the RFID tags within their coverage area and forward them the *reading request*. At this point, all RFID tags perform the step *QID generation*. Table 11.3 helps us to understand how $QIDs$ are generated by each monitored person: the second row of the table reports the initial value of ID associated with each person (i.e., $ID(0)$), whereas the third row reports the result of the application of the random permutation function (RPF) from step $t = 0$ to step $t = 1$, by assuming $g = 3$. The last row, instead, shows the final QID values for step $t = 1$ obtained as $ID(1)$ mod 2.

After this computation, every RFID tag sends the obtained QID to the RFID reader from which the *reading request* come. The readers collect the received $QIDs$ and process them. In particular, we assume that P_3 is under the coverage area of the reader R_1, P_4 is under the reader R_2, whereas P_1 and P_2 are under the reader R_3. Thus, both the readers R_1 and R_2 receive only one QID (from P_3 and P_4, respectively) and they send to the server the tuple $\langle QID, x \rangle$, where the first element is the QID received from the tag (0 in both cases) and the second element is the index of the reader (i.e., 1 and 2, respectively). As for the reader R_3, it receives the same QID from P_1 and P_2. For the first QID, the tuple $\langle 1, 3 \rangle$ is sent to the server, whereas the second QID is not directly sent to the server. Indeed, according to the step *Response*, in case of collision of some QID on the same reader, as it happens for P_1 and P_2, only one of them is associated with this reader, whereas the remaining $QIDs$ will be mapped to other readers. In particular, a random reader is selected (we assume it is the reader R_{87} in

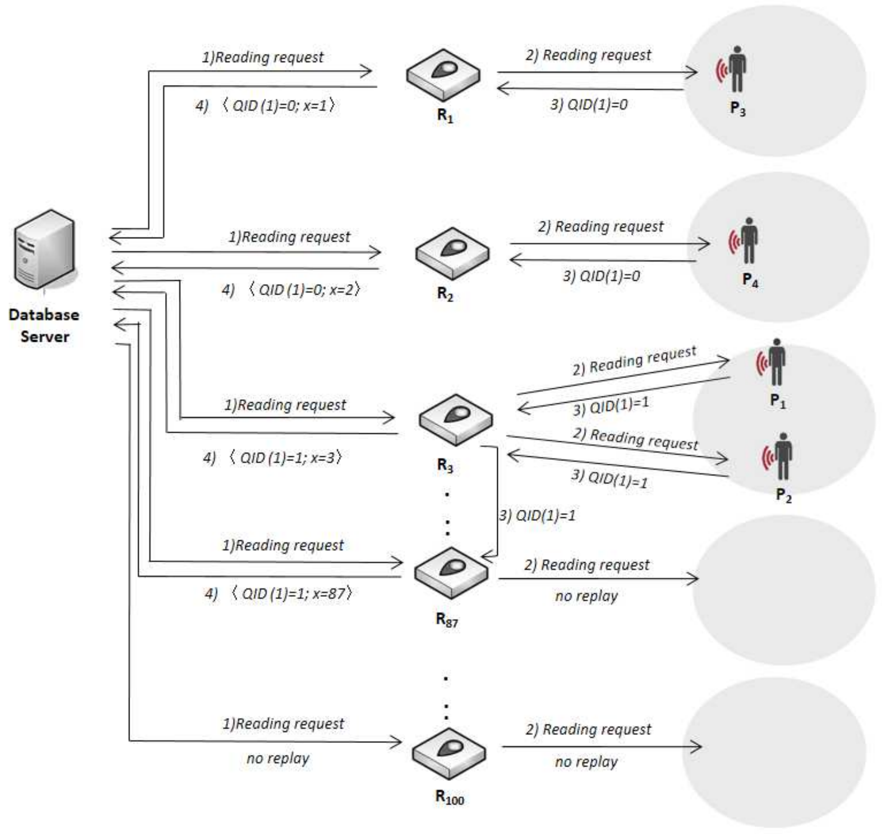


Fig. 11.1: An example of protocol instantiation.

Patients	P_1	P_2	P_3	P_4
$ID(0)$	1	2	3	4
$ID(1)$	3	1	4	2
$QID(1)$	1	1	0	0

Table 11.3: An example of the transformation of 4 tag identifiers.

our example) and the second QID is associated with this reader, in such a way to hide that more users with the same QID are in the same place (this could result in the violation of the privacy requirement). In practice, P_1 will be associated with reader R_3, whereas P_3 will be virtually mapped on the reader R_{87}. Finally, the server stores all the received records and checks that all expected QID are really received.

11.2 Case study

In this section, we describe the real-life scenario we considered in our experimental evaluation of the performance and security features of our approach. For this purpose, we referred to an existing ALF and built a simulator based on its physical characteristics. Specifically, the considered real-life ALF stretches over about 50k square meters and hosts a maximum number of 500 monitored residents. We designed our solution by computing the number of readers to cover all the different areas occurring in this facility and their exact position therein. Observe that the position of each reader depends on the power transmission level and the read distance has to guarantee that each portion of the place is covered by at least one reader.

To satisfy all the requirements of the techniques above for reader distribution, we have that the number of readers necessary to cover the entire area is 2000. Concerning the use of the RFID technology, RFID active tags and readers use an operating frequency of 433 MHz, which can be safely used also for healthcare applications. This is an important property as some of the ALF residents may suffer from important pathologies requiring severe constraints for the electrical equipments used for their care. The (adjustable) read range of such tags has been set to cover an area of about 40 meters (i.e., about a room). Each reader can detect hundreds of tags in few seconds and based on the signal strength received, the reader reports or ignores the received EPC to avoid multiple reading of the same tag (see Section 11.1). Tags have small size and are attached to person's wrist or ankle using standard ID straps.

This real-life scenario will be used in the next section as test bed for our experiments aimed to validate our proposal.

11.3 Security Model

In this section, we describe the security model and analyze the security properties of our proposal.

11.3.1 Attack Model

As usual in this context, we realistically assume that a solution satisfying the privacy requirement exists. Consequently, we do not consider inadmissible cases, for example with very few monitored people or in which k is too high with respect to the number of residents and the possible locations.

Under this basilar assumption, we state the security properties of our protocol. We recall that we aim to obtain *k-anonymity*, meaning that the probability of localizing a person is k^{-1}, where k is a given anonymity requirement. Our approach reaches this goal by forcing that more people generate the same QID.

We identify the following actors in our scenario:

- **service provider**, the entity that implements the RFID-based solution in the environment (i.e., installs and configures the readers, wires the connections among readers, provides the monitoring software, etc.).
- **residents**, who have to be monitored and whose privacy has to be preserved.
- **(system) administrators**, who can access all data produced by the service;
- **unmonitored people**, who are present in the environment but are not monitored (e.g., nurses or visitors).

Concerning the security features of our proposal, we identify the following properties that our approach must satisfy:

1. **SP1.** Given a QID q, at least k different readers report the presence of a resident sending q.
2. **SP2.** Resistance to attacks done by the administrators in which from the knowledge of the QID sent by a given resident the administrator wants to know his position.
3. **SP3.** Resistance to attacks based on the installation of fake tags by unauthorized people.
4. **SP4.** Resistance to client-side attacks based on eavesdropping.
5. **SP5.** Resistance to attacks based on the analysis of trajectories of residents.

To analyze the security properties above, our threat model includes the following assumptions. The security will be analyzed w.r.t. a parameter x (see below).

1. the service provider is trusted;
2. besides attacks considered by property SP3, no further physical attacks on the hardware infrastructure is possible;
3. the attacker cannot launch DoS attacks on tags, readers, and server;
4. the monitoring software run by tags, readers and server cannot be altered;
5. only passive attacks on the network traffic are possible;
6. the association between RFID tag and resident cannot be compromised;
7. the attacker has no knowledge about residents' habits;
8. the attacker is able to know the position of at most x residents.

Concerning assumptions 2 and 3, we note that their removal, does not give the attacker any information about residents' localization but results in the compromission of the effectiveness of the technique. Among all assumptions that are realistic, we observe that assumption 7 is commonly adopted in the literature [217].

Moreover, because compliance with **SP4** and **SP5** is strongly scenario-dependent, it would be infeasible to analytically prove it. Therefore, we will address this issue experimentally in a simulated real-life ALF. The real-life ALF we consider stretches over about 50k square meters. We set the number of monitored residents to 500 and we computed that the number of RFID readers necessary to cover the entire area is about 2000. Each reader can detect hundreds of tags in few seconds and based on the signal strength received, the reader reports or ignores the received EPC to avoid multiple reading of the same tag (see Section 11.1). Tags have small size and are attached to person's wrist or ankle using standard hospital ID straps. In our experiments, to simulate resident shifts inside a given area, we built a prototype implementing random-based mobility models described in [29]. The default model is that residents move randomly in the whole area.

11.3.2 Security Analysis

In this section, we study the security properties of our approach on the basis of the attack model described above. Recall that our target is to guarantee the privacy requirement k. First, we observe that our method is ϵ-approximate, because the privacy requirement is guaranteed with probability $1-\epsilon$, where ϵ is a small positive real. Moreover, the security is analyzed w.r.t. the parameter x defined earlier in Assumption 8. To do this, we just increase the privacy parameter k to the value $k+x$. Indeed, from the knowledge of the position of x residents, the attacker can guess the position of other residents with probability equal to $(k-x)^{-1}$ (thus the initial privacy requirement is satisfied).

Compliance with SP1. The next theorem proves that our proposal guarantees the security property **SP1** described in Section 11.3.1, i.e., given a QID q, it guarantees that at least k different readers report the presence of a resident sending q.

Theorem 11.1. Given an admissible privacy requirement k, there exist at least one configuration of the parameters d and a such that k-anonymity is guaranteed with probability $1-\epsilon$, for any $0<\epsilon<1$.

Proof. We need to compute two probabilities P_1 and P_2. For the first one, we introduce the random variable $\mathcal{C}_1$ defined as follows: given a QID q, $\mathcal{C}_1$ is the total number of residents who, at a given time t, generate q as QID. We call *number of QID-collisions* the value returned by $\mathcal{C}_1$. We study the probability $P_1(\mathcal{C}_1 = c_1)$, which is the probability to have exactly c_1 QID-collisions, with $0 \le c_1 \le p$. First, consider the case $a=1$. We obtain that $P_1(\mathcal{C}_1 = c_1) = 1$ if $c_1 = \frac{p}{d}$, $P_1(\mathcal{C}_1 = c_1) = 0$ otherwise. Indeed, when $a=1$, $QID(t)$ is equal to the deterministic value $\widetilde{QID}(t) \bmod d$ and the function RPF, which introduces pseudo-random permutations, guarantees that the possible QIDs are uniformly distributed among residents. Then, consider the case $a=0$. Now, the probability of having c_1 QID-collisions is: $P_1(\mathcal{C}_1 = c_1) = \frac{(d-1)^{p-c_1}\frac{p!}{(p-c_1)!}}{d^p}$. Indeed, d^p is the number of possible assignments of QIDs and $(d-1)^{p-c_1}\frac{p!}{(p-c_1)!}$ is the number of cases producing c_1 QID-collisions. Moreover, $P_1(\mathcal{C}_1 = c_1) = \frac{1}{d^{p-c_1}}\frac{1}{d^{c_1}}(d-1)^{p-c_1}\frac{p!}{(p-c_1)!} = \frac{p!}{(p-c_1)!}(\frac{1}{d})^{c_1}(1-\frac{1}{d})^{p-c_1}$. By combining the two cases above, we obtain:

$$P_1(\mathcal{C}_1 = c_1) = \begin{cases} a + (1-a)\frac{p!}{(p-c_1)!}(\frac{1}{d})^{c_1}(1-\frac{1}{d})^{p-c_1} \\ \qquad\qquad\qquad\qquad \textit{if } c_1 = \frac{p}{d} \\ \\ (1-a)\frac{p!}{(p-c_1)!}(\frac{1}{d})^{c_1}(1-\frac{1}{d})^{p-c_1} \\ \qquad\qquad\qquad\qquad \textit{otherwise} \end{cases}$$

The second probability we have to compute is related to the case in which, among the c_1 colliding residents, some of them are under the same reader. Consequently, the reader uses the function R (recall step *Response* in Section 11.1) to associate all (but the first one) colliding residents with other readers. However, it could happen that the new reader generates a *reader*-collision because another resident with the same QID is also under such a reader. Whereas QID-collisions help to increase privacy, reader-collisions are disadvantageous because more residents are associated with the same reader, thus reducing the number of possible locations for these residents. To study the second probability, we introduce the random variable $\mathcal{C}_2$ defined as the total number of times that the function R returns a reader that sends the same QID, thus generating a reader-collision. Therefore, we study the probability $P_2(\mathcal{C}_2^{c_1,c} = c_2)$, which is the probability that, having c_1 residents with the same QID and c of them under the same reader, we have c_2 cases in which the function R generates reader-collisions, with $0 \leq c_2 < c$. If we denote by R_A the set of readers associated with residents generating the same QID and by R_B the remaining ones, $P_2(\mathcal{C}_2^{c_1,c} = c_2)$ can be written as $\sum_{i=0}^{c_2} \Big(P_{2A}(\mathcal{C}_2^{c_1,c} = i) + P_{2B}(\mathcal{C}_2^{c_1,c} = c_2 - i) \Big)$, where the first term is the probability to have i cases in which the function R returns a reader in R_A (thus generating a reader-collision), whereas the second term is the probability to have $c_2 - i$ cases in which the reader-collision is obtained because the function R returns the same reader among those in R_B. The sum considers all the possible cases generating c_2 reader-collisions. $P_{2A}(\mathcal{C}_2^{c_1,c} = i) = i \cdot \frac{c_1 - c + 1}{r - 1}$ because we do i random generations, $|R_A| = c_1 - c + 1$, and the remaining readers are $r - 1$. Following the same reasoning seen for $P_1(\mathcal{C}_1 = c_1)$ in the case $a = 0$, we can write:

$$P_{2B}(\mathcal{C}_2^{c_1,c} = c_2 - i) = \frac{\Big(r - (c_1 - c + 1) - 1\Big)^{c - (c_2 - 1)} \frac{c!}{(c - (c_2 - i))!}}{r - (c_1 - c + 1)}.$$

The problem we have to solve is to find d and a such that, for a given (small) ϵ, the following inequality holds:

$$\sum_{c=k}^{p} \Big(P_1(\mathcal{C}_1 = c_1) + \sum_{i=0}^{c-k} P_2(\mathcal{C}_2^{c_1,c} = i) \Big) \geq 1 - \epsilon$$

In words, we require at least k residents generating the same QID (P_1). Then, if we have $k + x$ colliding residents, we accept also cases in which x are in the same place (the sum on P_2 deals with this possibility). By satisfying the above equation, we guarantee with probability $1 - \epsilon$ that at least k residents with the same QID in k different places occur, thus validating the theorem.

To conclude the proof, we show that, if a solution satisfying the privacy requirements exists (as stated in the hypothesis of the theorem), then at least one setting of d and a satisfying the above equation exists. Indeed, the trivial solution $d = 1, a = 1$ solves the above problem because all residents have the same QID. However, in this case, no meaningful information about the location of any resident is provided. As a consequence, we need to find the greatest d satisfying the above property. This number can be found by a guess-and-check iterative method, which starts from the minimum value $d = 2$ and at each iteration increases by 1 the value of d (as we will see below, the choice of a is related to some security considerations that suggest us to set $a = (1 - x^{-1})$, where x is a suitable positive integer). □

Compliance with SP2. In this case, we have to prove that our approach resists to attacks in which the administrator knows the QID sent by a given resident and wants to know his position. According to Theorem 11.1, with probability $1 - \epsilon$, an attacker finds that there are at least k different places in which the presence of a person with a given QID is reported. Consequently, no privacy leakage occurs.

Compliance with SP3. To be compliant with **SP3** our approach has to resist to attack in which fake tags are installed in the environment. We identify two cases: (i) an attacker introduces a fake tag to compromise the monitoring system; (ii) an attacker introduces a fake tag to allow a resident to leave the monitored environment.

In the first attack, the adversary simulates a tag and sends a random QID. First, we observe that the addition of a new tag is detected by the system, as $p + 1$ (instead of p) QIDs are received; Moreover, this does not violate residents' privacy because the QID generated by the attacker increases the number of collisions. As a follow up of this attack, in the second case, an adversary wants to allow a resident to leave the environment without being detected. However, to do this, the attacker has to know the whole sequence of QIDs generated by this resident. But this information is unknown.

Compliance with SP4.

To be compliant with **SP4** resistance to attacks based on eavesdropping must be guaranteed by our approach. Therefore, we consider the case in which an attacker knows the QID generated by a resident (for example, by eavesdropping the communication between tag and reader) and wants to guess who is the resident located near that reader at that time. However, residents' QIDs are dynamic and change at each reading in a pseudo-random way, so that no correlation between two subsequent readings should be guessed. Observe that, setting $a = 1$ means that the QID generation of each tag depends only on the random permutation function that guarantees a uniform distribution of QIDs and a number of collisions deterministically fixed. However, a side effect of setting $a = 1$ is that the sequence of QIDs generated by the same tag is periodic with period equal to the number of patients. An attacker may exploit this periodicity to know the next QIDs that a patient will generate and, thus, may try to track patient movements. The parameter a is used to prevent such an attack by randomly changing some elements of the deterministic QID sequence in such a way to break this periodicity. The parameter a measures the probability that an element of the deterministic QID sequence is not changed.

It is, then, obvious that the possibility to successfully carry out this attack depends on the values of the parameters of the system; therefore, by tuning such parameters, we can reduce the probability of success for this attack to any desired (low) value.

For this reason, we carried out several experiments to study the behavior of our technique for different values of system parameters. We start by studying the parameter a. In this first experiment, we choose five different values for a (i.e., 0.0, 0.1, 0.25, 0.5 and 1) and we measure the number C of colliding residents against the percentage d_C of the QIDs for which we observe C collisions. We set the parameter d (i.e., the number of possible QIDs) to $p/4$. In Figure 11.2, we report the results of this experiment.

From the analysis of this figure, we observe that the peak of each curve decreases if a assumes low values. In particular, if a is equal to 0, then our approach assigns QIDs in a pseudo-random way, according to the generator PRNG described in Section 11.1. For this reason, the trend of d_C is very smooth and the standard deviation of the number of collisions is high. By contrast, if a is equal to 1, then the QID generation is deterministic and, hence, we observe a very peaked trend of d_C for $C = 4$. Therefore, we should set a to the highest possible value allowing our approach to resist to attacks based on the periodicity of the QIDs

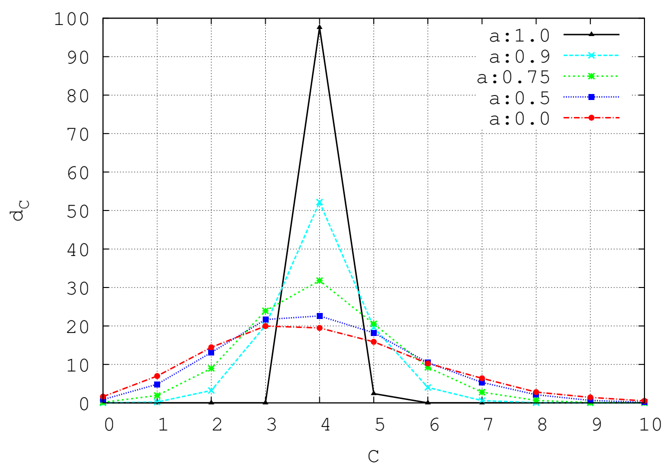


Fig. 11.2: Number of residents with the same QID versus d_C for different values of a.

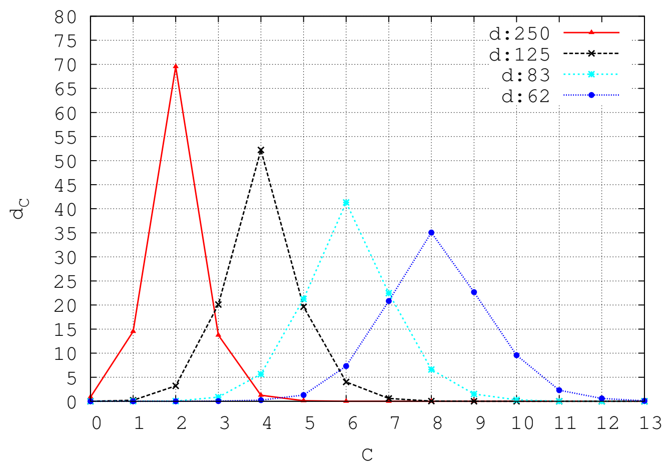


Fig. 11.3: Number of residents with the same QID versus d_C for different values of d.

generation. If we assume that changing one element every x elements of the deterministic sequence is sufficient to avoid a periodicity-based attack, we should set $a = (1 - x^{-1})$.

In the second experiment, we test the behavior of our approach for different values of the parameter d (i.e., $p/2$, $p/4$, $p/6$, and $p/8$), which identifies the number of different QIDs generated by residents. We set the value of the parameter a to 0.9. The result of this experiment is reported in Figure 11.3.

The discrete-Gaussian-like curves associated with the different values of d show decreasing height of the peak as d decreases, whereas the width of the bell (and, hence, the standard deviation) behaves the opposite. This can be explained by considering the fact that a lower value of d implies a greater number of users who generate the same QID. However, due to the presence of the parameter a, which represents the probability of assigning a QID different from that generated deterministically, the lower value of d also implies a greater number of residents that could have assigned a different QID w.r.t that assigned deterministically. These

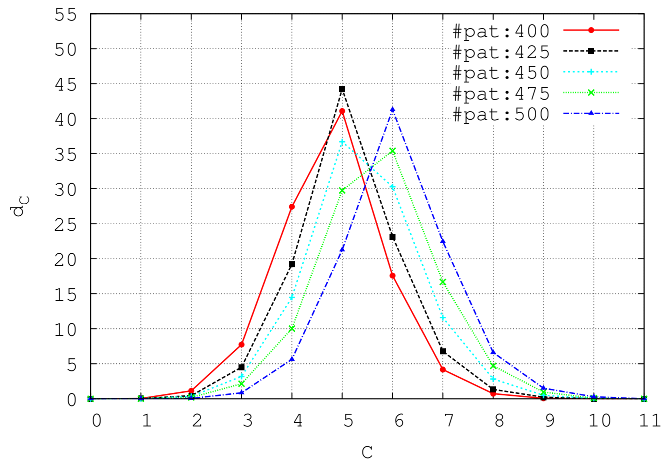


Fig. 11.4: Number of residents with the same QID versus d_C for different number of residents ($d = 83$).

residents will decrease the number of collisions for the QID they should have assigned deterministically and will increase the average number of collisions for the other QIDs. This causes the reduction of the height of the curve's peak and the increment of its standard deviation.

In the next experiment, we consider another issue. In a real-life scenario, it is possible that the number of residents may be less than the maximum number of residents (i.e., 500 in our experiments). However, once the RFID tags have been configured, the tag reconfiguration every time a resident arrives or leaves the ALF could be expensive. For this reason, we study the system performances when the number of residents is less than the expected one.

Figure 11.4 shows again the number C of colliding residents when the overall number of residents is decreased up to 20%. In this run, we set $a = 0.9$ and $d = 83$. Now, if we assume that the anonymity requirement is $k = 4$, then the anonymity requirement is not guaranteed in the worst case. This problem can be solved by reducing the value of d. A value of d reaching this goal can be found by applying a simple guess-and-check iterative method. For example, by changing the value d to 62, we obtain the results reported in Figure 11.5. With this new configuration, the anonymity requirement $k = 4$ is guaranteed with all the possible number of residents considered in this experiment.

Now, we compare the effectiveness of our technique in localizing a resident w.r.t. another approach that we call *naive*. In this approach, the tag of each resident is identified by a unique static ID. When a request for the resident with $ID = x$ is sent by readers, all tags act as follows. The tag with $ID = x$ replies to that request. The other tags generate a random r in the interval $[1, p]$, where p is the number of residents, and reply to that request if $r \leq k - 1$. Clearly, this approach expects that k tags reply to a location request on average, to satisfy the privacy requirement. In this experiment, we keep fixed the number of residents to 500 and we set $a = 0.9$ and $d = 125$ for our approach.

In Figure 11.6, we report the number C of residents' collisions measured for the two approaches. The result shows that our approach always outperforms the naive approach. Specif-

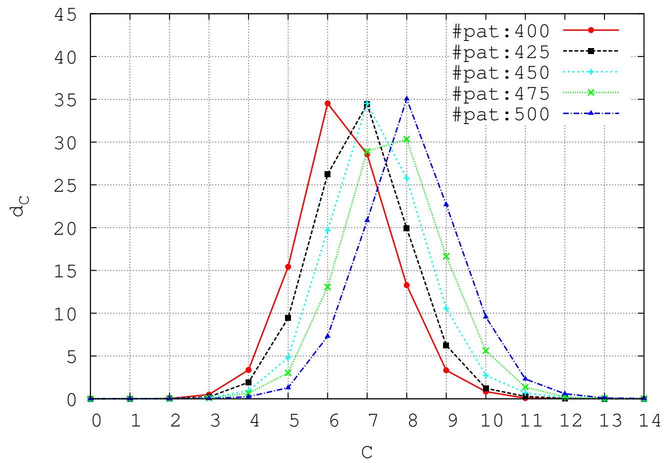


Fig. 11.5: Number of residents with the same QID versus d_C for different number of residents ($d = 62$).

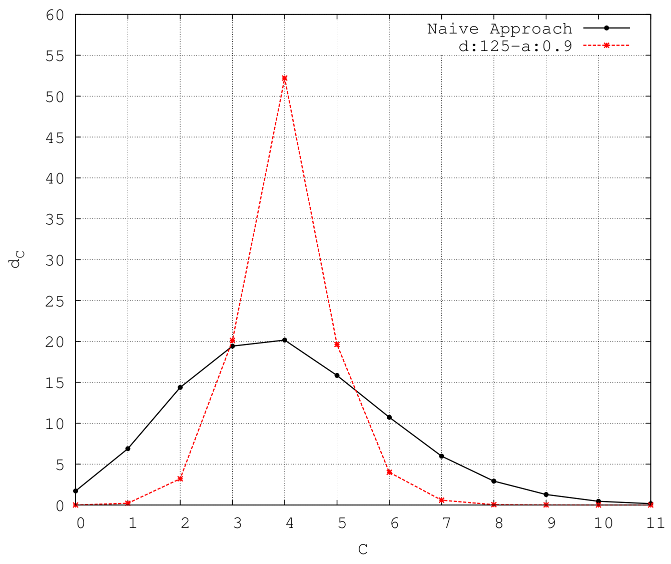


Fig. 11.6: Number of residents with the same QID versus d_C for the two techniques.

ically, it is worth noting that the curve associated with the latter has a trend similar to that obtained by our approach when $a = 1$.

Compliance with SP5. So far, we have considered a threat model in which the attacker eavesdrops the reading of one QID. As a consequence, we neglected the movements of residents, considering that all residents could be at any place of the environment at the reading time.

Now, to show the compliance of our approach with **SP5**, we consider the case in which the attacker eavesdrops consecutive reading of tag QIDs. In this new scenario, the frequency of tags reading assumes a great importance, as shown through the following example. If we consider an average speed of 1.6 m/s for each resident, we have that a resident takes about 11 minutes to move from any point to any point, because the maximum distance between two points inside the considered environment is about 1 kilometer. As a consequence, if all

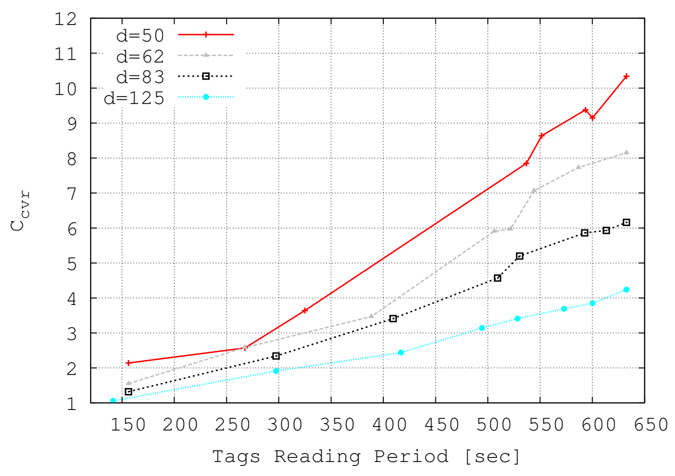


Fig. 11.7: Number of residents with the same QID inside a coverage area versus time (seconds).

readers query tags with a frequency higher than 1 reading every 11 minutes, then the attacker may take advantage from the reduced set of possible paths to guess residents' habits or, even worse, to identify them.

Therefore, to guarantee the compliance with **SP5**, we discuss the problem of setting a maximum reading frequency to solve the trade-off between privacy protection and localization accuracy. In the first experiment, we consider that each resident moves at a constant speed of 1.6 m/s and can choose a different movement direction (i.e., random shifts). We vary the reading period from 2.5 to 10.66 minutes because, as observed above, periods higher than 11 minutes do not give any advantage to the attacker. Now, given a resident u, we measure the number C_{cvr} of residents with the same QID who are inside the *coverage area* of u, where as coverage area we mean the area that u can cross in the reading period. In this experiment, we assume that the anonymity requirement is $k = 2$. In Figure 11.7, we report C_{cvr} versus tags reading period (in seconds) for different values of d.

Looking at the value $d = 125$, we observe that if we set the RFID-reader query time to 300 seconds (i.e., 5 minutes), then we guarantee that there are about two users inside a coverage area (i.e., the privacy requirement $k = 2$ is satisfied). Clearly, if the value chosen for d is lower than 125, then the query time can be reduced. For instance, by looking at the same figure, if $d = 50$, we find that a query time of 150 seconds is sufficient to satisfy the privacy requirement, because it assures at least two collisions (i.e., $C_{cvr} \geq 2$).

Concerning the influence of the average speed of residents on these results, we observe that the higher this speed, the wider the coverage area. As a consequence, an increase in the speed results in an improvement of the privacy degree obtained because a wider coverage area implies a higher probability of collisions of residents inside it. In practice, the effect of speed increasing is that all the curves of Figure 11.7 reach the final value (e.g., 4 for the curve $d = 125$) for a lower tag reading period.

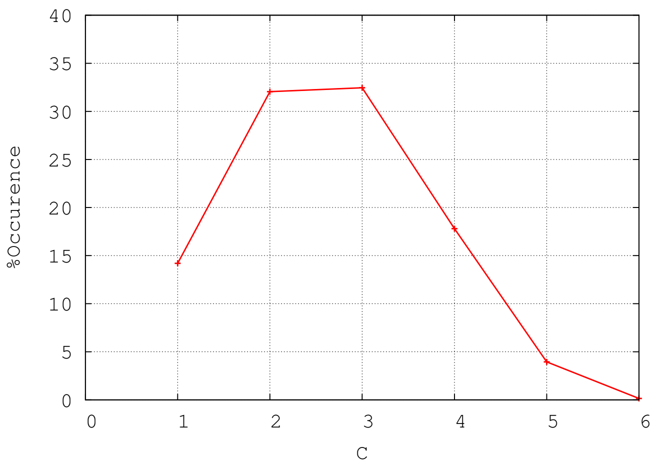


Fig. 11.8: Number of residents with the same QID inside the coverage area versus the percentage of involved users.

Now, we study possible attacks when residents' movements are not fully random, as actually happens in real world. For this purpose, we use as mobility model the Random Waypoint Model [29]. This mobility model generates resident shifts as described below. At the beginning of the simulation, each resident randomly selects one destination point inside the considered area. Then, he moves towards this end-point for a walking interval (W_{min}, W_{max}) with a speed selected uniformly at random in the interval (S_{min}, S_{max}). Speed and movement direction of each resident are selected independently of other residents. Once a resident reaches the end-point or ends his walking interval, he stops for a duration defined by the "pause" parameter that varies in the interval (P_{min}, P_{max}). After this interval, he continues his previous path or chooses another end-point and starts moving towards it. In our simulation, we set $W_{min} = 3.0$ seconds, $W_{max} = 10.0$ seconds, $S_{min} = 0.1$ m/s, $S_{max} = 1.6$ m/s, $P_{min} = 0.0$ seconds and $P_{max} = 6.0$ seconds.

In this experiment, we suppose the attacker is able to estimate the parameters of the residents' mobility model and to reduce the space of the possible target points. We set $a = 0.9$, $d = 125$, and the reading period to the value 300 seconds. For each resident u, we compute his coverage area as the maximum space he can cross in the tags reading period and count how many residents with the same QID as u fall in u's coverage area.

In Figure 11.8, we report the number of collisions measured inside the coverage area versus the percentage of the involved users.

This figure shows that 85% of residents share their QID with more than 2 other residents inside the coverage area. Only 14% of them are alone. Thus, the probability that a resident is alone inside the coverage area is 0.14, but the probability of being alone again at the next step is $0.14 * 0.14 = 0.019$ and becomes 0.003 if a third step is considered. Denoting by q the probability that a resident is alone inside a coverage area after one shift, an attacker guesses the i-th shift of a resident with a probability $\overline{q} = \prod_1^i p$. In practice, this experiment allows us to conclude that by suitably setting d, a, and the minimum tag reading period, we can reduce the probability q to satisfy any admissible privacy requirement.

11.4 Discussion

As a final discussion, we provide some considerations on the effectiveness of our service.

We may state the following:

- Concerning the use of the RFID technology, we briefly recall that RFID tags can be classified into passive, semi-active and active on the basis of computational power, presence of power supply, transmission range and cost. Moreover, it is possible to distinguish between two kinds of tags operating at different frequencies and, hence, having different and adjustable transmission range (from few meters to hundreds meters): UHF tags and HF tags. In our design, we adopt RFID active tags and readers operating at UHF frequency of 433 MHz, which is allowed for healthcare applications. It is worth noting that our solution is feasible from an economic point of view, as witnessed by the fact that commercial RFID-based solutions specifically oriented to track residents in assistive environments exist, even though they do not address at all privacy issues. In our case, the cost of the solution can be estimated by considering plausible prices of about \$2 per tag and \$200 – \$300 per reader. In addition, we have to consider the cost of the network infrastructure connecting the readers, even though also non-RFID solutions require (possibly more expensive) network infrastructures. On the other hand, it is known that RFID technology is more cost effective than other technologies for localization such as Bluetooth, WLAN or UWB (Ultra-wideband).
- The results obtained about the relationship between speed of residents (we consider 1.6 m/s) and parameter setting show that we obtain a good privacy level yet allowing quick residents' localization by means of a limited number of attempts and, thus, of human resources employed to manage emergency. This conclusion arises from the results of the experiments carried out to prove the compliance of our approach with **SP5** described in Section 11.3.2.
- The attack model is realistic because the environment cannot be considered closed as a relevant number of external persons (visitors, medical representatives, maintainers, cleaners, etc.) are often present in the ALF. Thus, the environment can be considered hostile.
- The need of free movement in the environment is particularly realistic as assisted living facility are used for people with disabilities in which resident activities are monitored to help to ensure their health, safety, and well-being.
- A limitation of our experimental study regards the environment, which is assumed to be a perfect rectangular. Clearly, the regular shape of the simulated area implies a uniform distribution of RFID readers. Actually, there could be some areas smaller or bigger, thus requiring to modify the coverage area set for each reader in the simulation. This approximation may lead to underestimate or overestimate the total number of RFID readers. However, these variations w.r.t. the real-life activity and the structure should not affect the conclusions arising from the analysis of the experimental results.

In conclusion, the approach presented in this chapter allows us to localize residents in an assisted living facility by preserving their privacy. As a matter of fact, even though the party that performs data elaboration and administration (e.g., medical staff, nursery managers, IT staff, etc.) can be assumed trusted, it is not true, in general, that the utility of having precise information about residents' location is stronger than the right of keeping private the exact movements of residents. The solution of the above trade-off is the main added value our work w.r.t. the existing literature, in which no protection against administrator-side attacks is provided. Our study, shows that the proposed service is enough flexible and precise to be an effective tool for residents' localization. Moreover, we show that the method is robust against several possible attacks on privacy.

11.5 Related Work

In this section, we survey the literature regarding location anonymity, finally focusing on health-care environments.

Generally, different applications imply different privacy threats and models. For instance, in mobile phones, LBS applications using Global Positioning System (GPS) [105], or Wireless Networks, all moving nodes have continuous timestamps. Therefore, the user's location is continuously detected [224, 22]. This assumption, instead, does not hold for an RFID scenario, because an RFID-tagged object (e.g., a smart card) is unlikely to be continuously detected. In this case, high computational efficiency and low cost is needed, moreover the approach presented in [249] can be used to improve the efficiency of the RFID system. In this context, an approach to tracking children in a large park still preserving their privacy is presented in [163]. Through a deep security analysis the authors show that their tracking scheme guarantees children identity privacy, unlinkable location privacy, and forward security. However, this approach does not contrast server side attacks. This limitation is not present in our approach.

The problem of balancing (1) the need of protecting health information and (2) the utility of sharing information, has received great attention in the last years [168, 240].

RFID technology and data anonymization have been widely adopted in health-care environments [226, 81]. According to the results presented in [234], we propose a practical solution to the problem of localizing people in an assistive environment, to reduce risks related to hospitalization and to increase patient independence. This feature, together with the others described above (i.e., no third trusted party used, no obfuscation, administrator-side attack prevention), makes our approach relevant and novel with respect to the state of the art.

Part III

Final Issues

This part of the thesis is devoted to point out some final remarks and possible extensions of the topics presented previously. In particular, in Chapter 12 we have a look at some future research, whereas in Chapter 13 we draw our conclusions. In addition, in this part, we include a bibliography indicating both the papers presented in the literature related to the topics this thesis focuses on, and our contributions.

12

Future Directions

In this chapter, we describe some future developments of our research. In particular, in the first section, we describe some future directions in the field of privacy, security and trust in online communities. Whereas, in the last section, we focus on some future works dealing with privacy and security on both public and private physical organizations.

12.1 Online Communities

As a first task in the context of online communities, we have implemented a model to generalize and match concepts, actions and relationships of existing social networks. This has been done to uniformly handle social network data and as a mean to accomplish the crucial issue of crawling data from a multi-social network scenario to retrieve all the information needed for our study on privacy and security in OSNs. We plan to go further along this path and to design a multi-social network oriented language based on our model.

This language could help us to extend also the approach presented in Chapter 4. Here, we described a middleware to allow fine-grained access control of Twitter applications. This solution could be developed also for other mobile platforms and social networks. Still in this context, another direction we plan to follow is the definition of both a specific access control model and an anomaly detection module, which, till now, we have considered as orthogonal problems. The first module will extend the study described in this thesis by providing an attribute-based model for access control rules. Whereas, the anomaly detection module will have to implement a further security mechanism allowing the detection of malicious behavior of third-party applications.

As for trust in OSNs, an interesting future direction is the definition of trust models for the certification of digital identities in social networks. This objective can be reached by combining structural information from social networks with biometric data recorded during the profile registration phase and constantly monitored as users interact with social network services. Specifically, we consider chains of trust among users, and each chain starts from a root profile, which is a profile certified by a Profile Certification Authority. To build a certified profile (or root profile), a user has to register to the social network via a Profile Certification Authority (also by exchanging of identification documents). In this phase, the Profile Certification Authority gathers the biometrical parameters of the user to create a model that will be exploited, in the future interaction with him, to verify whether the account is still under control of this user. In the negative case, the profile will be no longer certified. Starting from a root node a trust chain is built by considering his neighborhoods. Specifically, the root profile directly certifies his 0-level neighborhood. For this purpose, we consider a preliminary (safe) step in which the root profile builds a biometric model of his direct contacts (this can be done, for instance, during first messaging interactions). After this preliminary step, the root profiles will continue to monitor his 0-level neighbors and will compute their direct trust level based on the comparison between their current biometric behavior and that recorded during the preliminary step. This procedure, can be now reiterated by 0-level neighborhood profiles with their contacts (1st-level neighborhood of root profile), thus creating a chain of trust relations.

The following example clarifies the protocol sketched above.

The user A wants to get in contact with a user B; therefore he uses social network features to find the account of B. To be sure that the account he is contacting really belongs to B, he will compute a trust value for this account by applying the following algorithm:

- if B is a root profile, then the maximum value of trust is obtained;
- an indirect trust value will be computed, otherwise.

The indirect trust level, is computed by propagating the direct trust values on the friendship graph of the social network. Specifically, we may consider applying a PageRank-based algorithm to evaluate the indirect trust level of profiles. In this case, each profile will receive a boost in their trust level based on the trust level of their neighbors.

12.2 Physical Organizations

In the context of secure transactions, a lot of future directions can be thought. In particular, in Chapter 8 the model `Tweetchain` is proposed as an alternative public ledger that ensures transaction security by building a meshed chain of tweets. A number of possible applications can be implemented over `Tweetchain` related to e-voting, e-commerce, document exchange, ticketing, crowdshipping, reputation systems, tourism, advertising, etc. Among these, the implementation of the paradigm of smart contracts over `Tweetchain`, could be of particular importance. Indeed, this application is currently thought for Blockchain 2.0 which allows the execution of code, written in a Turing-complete language, inside transactions. Therefore, we plan to improve `Tweetchain` by introducing the support to a Touring-complete language for transactions.

In the context of urban security, instead, we plan to focus our research on possible applications of the Internet of Things (IoT) [58]. This new scenario represents a good meeting point between social communities and physical organizations. Indeed, the information coming form OSN profiles could be used to improve some aspects of the physical networks of the objects belonging to the people who own these profiles. In particular, we plan to study a new selection criterium of links among objects to guarantee an effective access to services and data in a network of things.

Accordingly to the Internet of Things paradigm the establishment of new connections among nodes is driven by the discovery of the availability of desirable services or matching devices. Typically, the fact that two objects get in touch somewhere, sometimes (maybe because the corresponding owners meet in a certain location) is enough to trigger (with a given threshold) the establishment of a link between the two objects. This property is called *proximity*. Our aim will be that to identify possible enhanced ways to discover potentially beneficial links.

To do this, preliminarily, we need to identify the properties that can be used to build a more complex model.

In the literature, such properties have been identified as: *(i)* proximity, *(ii)* homogeneity, i.e., they are the same kind of object created by the same manufacturer; *(iii)* ownership, i.e., they belong to the same user; *(iv)* friendship, i.e., owners are mutual friend in a social network.

Arguing that decisions regarding the formation of new links among objects could rely on a mix of the above properties, we define a decision function to establish if a link between two objects $\langle x, y\rangle$ has to be created or not.

Observe that all the above properties give us some information about the direct relationship between two objects. We argue that also some indirect knowledge coming from the social networks of owners could be of high importance to support the decision function. Actually, having an indirect knowledge that can be used as a filter of the quality of links, our intention is to use all the above direct properties in addition to the classical selection criterium based on the sole proximity. Therefore, we introduce two measures, which we combine to compute the aimed decision function. These are: $T_{x,y}^{dir}$, which derives from the *direct* knowledge about objects and owners, and $T_{x,y}^{ind}$, which encodes some *indirect* knowledge.

As indirect knowledge, we plan to use interest assortativity [51], by means of which it is possible to have a measure of the correlation between a given human interest and the presence of links between humans. This choice is motivated by observing that the creation of a new link between two objects should also depend on the contexts in which these objects are used by the corresponding owners. Therefore, the information about interest assortativity may help to establish the level of membership of an object to a given context.

13

Conclusions

In this thesis, we have presented several activities performed in the context of trust, security and privacy within communities. Specifically, the contribution provided by this thesis concerns two referring macro-areas, namely *(i)* online communities, and *(ii)* physical organizations.

As for the research activities related to the first macro-area, we started by describing a multi-social network model and a system to provide meta-APIs. These tools have been thoroughly exploited in this thesis to extract and handle heterogeneous information from this complex scenario. Then, we have analysed the behavior of OSN users when it comes of privacy and disclosure of personal information, friendship and activity level. In this analysis, we posed particular attention on how users perceive privacy and on the robustness of Facebook privacy settings. We continued this activity by focusing on a close subject which is acquiring an always increasing importance, i.e., the security of mobile applications exploiting social network services. In this context, we proposed a middleware approach to allow the definition of fine-grained access control policies of applications using social network APIs. Finally, we discussed the topic of trust in online communities and proposed a reputation model that considers service providers, users and feedbacks, to implement the theoretical notion of certified reputation. Through this model we have concretely defined a strategy to *normalize* feedback scores towards reliable values in reputation systems.

Concerning the research activities dealing with the second macro-area, we proposed some approaches focused on privacy and security in the contexts of both public and private physical organizations. Specifically, in the field of e-government we presented a new electronic signature protocol that does not use public-key encryption, qualified signature creation devices, or qualified certificates. Then, we proposed an authentication scheme supporting full anonymity of users and unlinkability of service requests in the cloud. Finally, we presented an alternative to Blockchain leveraging services from social networks to ensure transaction security among untrusted parties. We implement this idea in an approach, called `TweetChain`, which uses services from Twitter to reach the above goal.

After these studies, we moved to some approaches whose application lays in the context of urban security. In particular as a first issue, we observed that video-surveillance is becoming more and more pervasive in people daily lives. To handle the huge size of generated data a lot of public and private organizations adopt cloud solutions for the storage. This may introduce risks if cloud providers are not trusted. Therefore, we proposed an approach allowing the user to verify that query results are *complete* (i.e., no qualifying tuples are omitted), *fresh* (i.e., the newest version of the results are returned), and *correct* (i.e., the result values are not corrupted).

Finally, still in the context of people monitoring, we designed a complex framework that allows the tracing of people movements using a privacy-preserving identification technique. We described the application of this system to two interesting scenarios: the first is related to critical infrastructures, whereas the second to Assistive Living Facilities. We showed the effectiveness of our solution through a deep experimental campaign for both scenarios.

Ringraziamenti

Il mio percorso da studentessa è giunto al termine. Al traguardo non posso che pensare ai miei cari che mi hanno accompagnato durante questo viaggio con pazienza e amore. È a tutti loro che dedico questo lavoro di tesi. Ringrazio di cuore ogni componente della mia grande famiglia per la presenza e il supporto. Prima di concludere, però, vorrei rivolgermi a qualcuno in particolare.

A mia zia Giusy va il mio pensiero più tenero. Le auguro una rinascita e prego che il suo sorriso dolce torni presto a risplendere in mezzo a noi anche più bello di prima.

A mia nonna. Perchè il suo affetto e la sua forza sono per me punto di riferimento costante e un rifugio sicuro.

Ai miei nonni che non ci sono più, ma che sarebbero stati più che fieri dei miei risultati.

Alla mia seconda famiglia, perchè in loro ho trovato un padre, una madre e una sorella che mi amano incondizionatamente, e questa è per me una delle più grandi fortune.

A Felicia, Filippa e Chad perchè mi donano ogni giorno nuove prospettive da cui osservare il mondo.

Al mio fidanzato a cui non riservo solo un ringraziamento, ma la promessa di ringraziarlo tutti i giorni della mia vita.

E, infine, a mio padre, mia madre e mio fratello. Sono le fondamenta del mio essere e non ho parole per esprimere loro il mio amore, ma so che il miglior modo per ringraziarli, dovunque il futuro mi porterà, sarà quello di essere felice.

References

1. Directive 99/93/CEE. http://eur-lex.europa.eu/legal-content/EN/ALL/;jsessionid=TCsMT1yBQ965GRJTMG9GnFDxQqYP1W7Y1LFLLkwsmjvWRy1Q15FJ!527097711?uri=CELEX:31999L0093.
2. Twitter. https://dev.twitter.com/docs, 2012.
3. Method and apparatus for monitoring movements of an individual. http://www.google.com/patents/US6049281, 2014.
4. OpenCV Website. http://opencv.org/, 2014.
5. RFID Adaptor Board Website. http://beagleboard.org/project/RFIDADP/, 2014.
6. Android Developers. https://developer.android.com/index.html, 2015.
7. Android Internal Storage. https://developer.android.com/guide/topics/data/data-storage.html#filesInternal, 2015.
8. Twitter authentication. https://support.twitter.com/articles/20171580, 2015.
9. Twitter.com API Documentation. https://dev.twitter.com/overview/documentation, 2015.
10. bitShark. https://play.google.com/store/apps/details?id=blake.hamilton.bitshark, 2016.
11. Common Criteria. http://www.commoncriteriaportal.org/cc/, 2016.
12. DroidWall. https://code.google.com/p/droidwall/, 2016.
13. EIDAS. http://eur-lex.europa.eu/legal-content/IT/TXT/PDF/?uri=CELEX:32014R0910&from=DE, 2016.
14. Facebook Graph API Documentation. https://developers.facebook.com/docs/graph-api, 2016.
15. Firewall analyzer. https://www.manageengine.com/products/firewall/employee-internet-monitoring.html, 2016.
16. Firewall pk+. https://play.google.com/store/apps/details?id=com.ikramshah.firewallpk, 2016.
17. Mobile security and antivirus. https://play.google.com/store/apps/details?id=com.avast.android.mobilesecurity, 2016.
18. Network Log. https://play.google.com/store/apps/details?id=com.googlecode.networklog, 2016.

19. On Electronic Identification and Trust Services for Electronic Transactions in the Internal Market and Repealing Directive 1999/93/EC. http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2014.257.01.0073.01.ENG, 2016.
20. Security SSL. http://developer.android.com/training/articles/security-ssl.html#Concepts, 2016.
21. Michael Abd-El-Malek, Gregory R Ganger, Garth R Goodson, Michael K Reiter, and Jay J Wylie. Fault-scalable byzantine fault-tolerant services. In *ACM SIGOPS Operating Systems Review*, volume 39, pages 59–74. ACM, 2005.
22. O. Abul, F. Bonchi, and M. Nanni. Never walk alone: Uncertainty for anonymity in moving objects databases. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 376–385. Ieee, 2008.
23. O. Abul, F. Bonchi, and M. Nanni. Anonymization of moving objects databases by clustering and perturbation. *Information Systems*, 35(8):884–910, 2010.
24. Berker Agir, Thanasis G Papaioannou, Rammohan Narendula, Karl Aberer, and Jean-Pierre Hubaux. User-side adaptive protection of location privacy in participatory sensing. *Geoinformatica*, 18(1):165–191, 2014.
25. Nadeem Akhtar. Social network analysis tools. In *Communication Systems and Network Technologies (CSNT), 2014 Fourth International Conference on*, pages 388–392. IEEE, 2014.
26. ALFAA. Assisted living facilities association of america. *An overview of the assisted living industry*, 1993.
27. Giuseppe Ateniese, Carlo Blundo, Alfrede De Santis, and Douglas R Stinson. Constructions and bounds for visual cryptography. In *Automata, Languages and Programming*, pages 416–428. Springer, 1996.
28. Adam Back, Matt Corallo, Luke Dashjr, Mark Friedenbach, Gregory Maxwell, Andrew Miller, Andrew Poelstra, Jorge Timón, and Pieter Wuille. Enabling blockchain innovations with pegged sidechains. *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, 2014.
29. F. Bai and A. Helmy. A survey of mobility models. *Wireless Adhoc Networks. University of Southern California, USA*, 206, 2004.
30. Jöran Beel and Bela Gipp. Enhancing search applications by utilizing mind maps. In *Proceedings of the 21st ACM Conference on Hypertext and Hypermedia*, pages 303–304. ACM, 2010.
31. István Zsolt Berta, Levente Buttyán, and István Vajda. Mitigating the untrusted terminal problem using conditional signatures. In *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, volume 1, pages 12–16. IEEE, 2004.
32. Angela Bohn, Christian Buchta, Kurt Hornik, and Patrick Mair. Making friends and communicating on facebook: Implications for the access to social capital. *Social Networks*, 37:29–41, 2014.
33. Booking.com. The Booking Site. http://www.booking.com/, 2015.

34. Tuhin Borgohain, Uday Kumar, and Sugata Sanyal. Survey of security and privacy issues of internet of things. *arXiv preprint arXiv:1501.02211*, 2015.

35. Janez Brank, Marko Grobelnik, and Dunja Mladenić. A survey of ontology evaluation techniques. In *In Proceedings of the Conference on Data Mining and Data Warehouses (SiKDD 2005)*, 2005.

36. Francesco Buccafurri, Gianluca Caminiti, and Gianluca Lax. Fortifying the dalì attack on digital signature. In *Proceedings of the 2nd International Conference on Security of Information and Networks*, pages 278–287. ACM, 2009.

37. Francesco Buccafurri, Vincenzo Daniele Foti, Gianluca Lax, Antonino Nocera, and Domenico Ursino. Bridge analysis in a social internetworking scenario. *Information Sciences*, 224:1–18, 2013.

38. Francesco Buccafurri, Lidia Fotia, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. A lightweight electronic signature scheme using Twitter. In *Proc. of the Italian Symposium on Advanced Database Systems (SEBD 2015)*, pages 160–167, Gaeta, IT, 2015.

39. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. A System for Privacy-Preserving Access Accountability in Critical Environments. *Journal of Pervasive and Mobile Computing*. Currently under Revision.

40. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. A Model to Support Multi-Social-Network Applications. In *Proc. of the International Conference Ontologies, DataBases, and Applications of Semantics (ODBASE 2014)*, pages 639–656, Amantea, Italy, 2014. Springer.

41. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. A Privacy-Preserving Solution for Tracking People in Critical Environments. In *Proc. of the International Workshop on Computers, Software & Applications (COMPSAC'14)*, pages 146–151, Västerås, Sweden, 2014. IEEE Computer Society.

42. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. Fortifying tripadvisor against reputation-system attacks. In *Internet Security (WorldCIS), 2014 World Congress on*, pages 20–21. IEEE, 2014.

43. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. Generating K-Anonymous Logs of People-Tracing Systems in Surveilled Environments. In *Atti del Ventiduesimo Convegno Nazionale su Sistemi Evoluti per Basi di Dati (SEBD'14)*, pages 37–44, Sorrento Coast, Italy, 2014.

44. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. A Model Implementing Certified Reputation and its Application to TripAdvisor. In *Proc. of the International Conference on Availability, Reliability and Security (ARES 2015)*, pages 218–223, Touluse, France, 2015. IEEE.

45. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. Accountability-Preserving Anonymous Delivery of Cloud Services. In *Proc. of the International Conference on Trust, Privacy and Security in Digital Business (TRUSTBUS 2015)*, pages 124–135. Springer, 2015.

46. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. Comparing twitter and facebook user behavior: privacy and other aspects. *Computers in Human Behavior*, 52:87–95, 2015.
47. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. A Middleware to Allow Fine-Grained Access Control of Twitter Applications. In *Proc. of the international conference on mobile, secure and programmable networking (MSPN 2016)*, pages 168–182, Paris, France, 2016. Springer.
48. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. A Privacy-Preserving Localization Service for Assisted Living Facilities. *IEEE Transaction on Service Computing*, 2016. In Press.
49. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. A threat to friendship privacy in Facebook. In *Proc. of the International Cross Domain Conference and Workshop (CD-ARES 2016)*, pages 96–105, Salzburg, Austria, 2016. Springer.
50. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. Completeness, Correctness and Freshness of Cloud-Managed Data Streams. In *Proc. of the Italian Symposium on Advanced Database Systems (SEBD 2016)*, pages 134–141, Lecce, IT, 2016.
51. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. Interest Assortativity in Twitter. In *Proc. of the International Conference on Web Information Systems and Technologies (Webist 16))*, volume 1, pages 239–246, Rome, Italy, 2016. SCITEPRESS.
52. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. A model to support design and development of multiple-social-network applications. *Information Sciences*, 331:99–119, 2016.
53. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. Range Query Integrity in Cloud Data Streams with Efficient Insertion. In *Proc. of the 15th International Conference on Cryptology and Network Security (CANS 2016)*, pages 719–724, Milan, Italy, 2016. Springer.
54. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. Range Query Integrity in the Cloud: the Case of Video Surveillance. In *Proc. of the International Conference for Internet Technology and Secured Transactions (ICITST-2016)*, Barcelona, Spain, 2016. IEEE.
55. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. Range Query Integrity in Cloud Data Streams with Efficient Insertion. In *Proc. of the 1st Italian Conference on Cybersecurity (ItaSec2017)*, Venice, Italy, 2017.
56. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. Tweetchain: An alternative to blockchain using Twitter. In *Proc. of the International Conference on ICT Systems Security and Privacy Protection (IFIP SEC 2017)*, Rome, Italy, 2017. Submitted for publication.
57. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, and Antonino Nocera. Using Twitter for Consensus: Tweetchain as Alternative to Blockchain. In *Proc. of the International Conference on Web Engineering (ICWE 2017))*, Rome, Italy, 2017. Submitted for publication.

58. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, Antonino Nocera, Luca Console, and Assunta Matassa. Twitter Interest Assortativity and its Application to Privacy and the Internet of Things. *International Journal of Human-Computer Studies*. Currently Under Revision.
59. Francesco Buccafurri, Gianluca Lax, Serena Nicolazzo, Antonino Nocera, and Lidia Fotia. A new approach for electronic signature. In *Proc. of the International Conference on Information Systems Security and Privacy (ICISSP 16))*, pages 440–447, Rome, Italy, 2016. SCITEPRESS.
60. Francesco Buccafurri, Gianluca Lax, and Antonino Nocera. A new form of assortativity in online social networks. *International Journal of Human-Computer Studies*, 80:56–65, 2015.
61. Francesco Buccafurri, Gianluca Lax, Antonino Nocera, and Domenico Ursino. Discovering links among social networks. In *Machine Learning and Knowledge Discovery in Databases*, pages 467–482. Springer, 2012.
62. Francesco Buccafurri, Gianluca Lax, Antonino Nocera, and Domenico Ursino. Moving from social networks to social internetworking scenarios: The crawling perspective. *Information Sciences*, 256:126–137, 2014.
63. Francesco Buccafurri, Gianluca Lax, Antonino Nocera, and Domenico Ursino. Discovering missing me edges across social networks. *Information Sciences*, 319:18–37, 2015.
64. Francesco Buccafurri, Gianluca Lax, Antonino Nocera, and Domenico Ursino. A system for extracting structural information from social network accounts. *Software: Practice and Experience*, 45(9):1251–1275, 2015.
65. Moira Burke, Cameron Marlow, and Thomas Lento. Feed me: motivating newcomer contribution in social network sites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 945–954. ACM, 2009.
66. William E Burr, Donna F Dodson, and William T Polk. *Electronic authentication guideline*. NIST 800-63-2, 2006.
67. Carol C Burt, Barrett R Bryant, Rajeev R Raje, Andrew Olson, and Mikhail Auguston. Model driven security: unification of authorization models for fine-grain access control. In *Enterprise Distributed Object Computing Conference, 2003. Proceedings. Seventh IEEE International*, pages 159–171. IEEE, 2003.
68. G. Caldarelli. *Scale-Free Networks: Complex Webs in Nature and Technology*. Number 9780199211517 in OUP Catalogue. Oxford University Press, 2007.
69. F. Carmagnola and F. Cena. User identification for cross-system personalisation. *Information Sciences*, 179(1-2):16–32, 2009.
70. Luca Caviglione, Jean-Francois Lalande, Wojciech Mazurczyk, and Steffen Wendzel. Analysis of human awareness of security and privacy threats in smart environments. *arXiv preprint arXiv:1502.00868*, 2015.
71. David Chaum and Sandra Roijakkers. Unconditionally-secure digital signatures. In *Advances in Cryptology-CRYPT0ï£¡90*, pages 206–214. Springer, 1991.
72. David Chaum and Eugène Van Heyst. Group signatures. In *Advances in Cryptology–EUROCRYPT’91*, pages 257–265. Springer, 1991.

73. Xu Cheng, Cameron Dale, and Jiangchuan Liu. Statistics and social network of youtube videos. In *Quality of Service, 2008. IWQoS 2008. 16th International Workshop on*, pages 229–238. IEEE, 2008.
74. Christy MK Cheung, Pui-Yee Chiu, and Matthew KO Lee. Online social networks: Why do students use facebook? *Computers in Human Behavior*, 27(4):1337–1343, 2011.
75. R. Chow and P. Golle. Faking contextual data for fun, profit, and privacy. In *Proceedings of the 8th ACM workshop on Privacy in the electronic society*, pages 105–108. ACM, 2009.
76. Sherman SM Chow, Yi-Jun He, Lucas CK Hui, and Siu Ming Yiu. Spice–simple privacy-preserving identity-management for cloud environment. In *Applied Cryptography and Network Security*, pages 526–543. Springer, 2012.
77. Simone Cirani, Marco Picone, Pietro Gonizzi, Luca Veltri, and Giorgio Ferrari. Iot-oas: An oauth-based authorization service architecture for secure services in iot scenarios. *Sensors Journal, IEEE*, 15(2):1224–1234, 2015.
78. Dwaine Clarke, Blaise Gassend, Thomas Kotwal, Matt Burnside, Marten Van Dijk, Srinivas Devadas, and Ronald Rivest. The untrusted computer problem and camera-based authentication. In *Pervasive Computing*, pages 114–124. Springer, 2002.
79. Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Designing Privacy Enhancing Technologies*, pages 46–66. Springer, 2001.
80. Mauro Conti, Vu Thien Nga Nguyen, and Bruno Crispo. Crepe: Context-related policy enforcement for android. In *Information Security*, pages 331–345. Springer, 2011.
81. William Lee Croft, Wei Shi, Jorg-Rudiger Sack, and Jean-Pierre Corriveau. Geographic partitioning techniques for the anonymization of health care data. *arXiv preprint arXiv:1505.06786*, 2015.
82. Sabrina De Capitani di Vimercati, Sara Foresti, Sushil Jajodia, Stefano Paraboschi, and Pierangela Samarati. Integrity for join queries in the cloud. *Cloud Computing, IEEE Transactions on*, 1(2):187–200, 2013.
83. Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*, pages 3–18. Springer, 2015.
84. Peter J Denning. Fault tolerant operating systems. *ACM Computing Surveys (CSUR)*, 8(4):359–389, 1976.
85. Premkumar Devanbu, Michael Gertz, Charles Martel, and Stuart G Stubblebine. Authentic third-party data publication. In *Data and Application Security*, pages 101–112. Springer, 2001.
86. Premkumar Devanbu, Michael Gertz, Charles Martel, and Stuart G Stubblebine. Authentic data publication over the internet. *Journal of Computer Security*, 11(3):291–314, 2003.
87. Ratan Dey, Zubin Jelveh, and Keith Ross. Facebook users have become much more private: A large-scale study. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012 IEEE International Conference on*, pages 346–352. IEEE, 2012.

88. Claudia Diamantini, Domenico Potena, and Emanuele Storti. Semantically-supported team building in a kdd virtual environment. In *Collaboration Technologies and Systems (CTS), 2012 International Conference on*, pages 45–52. IEEE, 2012.
89. R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.
90. RIM Dunbar. Social cognition on the internet: testing constraints on social network size. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1599):2192–2201, 2012.
91. Cynthia Dwork. Differential privacy: A survey of results. In *Theory and Applications of Models of Computation*, pages 1–19. Springer, 2008.
92. Catherine Dwyer, Starr Hiltz, and Katia Passerini. Trust and privacy concern within social networking sites: A comparison of facebook and myspace. *AMCIS 2007 Proceedings*, page 339, 2007.
93. Nicole B Ellison, Charles Steinfield, and Cliff Lampe. The benefits of facebook "friends:" social capital and college students' use of online social network sites. *Journal of Computer-Mediated Communication*, 12(4):1143–1168, 2007.
94. William Enck, Machigar Ongtang, and Patrick McDaniel. On lightweight mobile phone application certification. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 235–245. ACM, 2009.
95. P. Erdös and A. Rényi. On Random Graphs, I. *Publicationes Mathematicae*, 6:290–297, 1959.
96. Ittay Eyal. The miner's dilemma. In *2015 IEEE Symposium on Security and Privacy*, pages 89–103. IEEE, 2015.
97. Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert Van Renesse. Bitcoin-ng: A scalable blockchain protocol. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pages 45–59, 2016.
98. Pietro Ferrara, Omer Tripp, and Marco Pistoia. Morphdroid: Fine-grained privacy verification. In *Proceedings of the 31st Annual Computer Security Applications Conference*, pages 371–380. ACM, 2015.
99. BJ Fogg and Daisuke Iizawa. Online persuasion in facebook and mixi: A cross-cultural comparison. In *Persuasive technology*, pages 35–46. Springer, 2008.
100. P.A. Fouque, G. Poupard, and J. Stern. Sharing decryption in the context of voting or lotteries. In *Financial Cryptography*, pages 90–104. Springer, 2001.
101. Adam P Fuchs, Avik Chaudhuri, and Jeffrey S Foster. Scandroid: Automated security certification of android applications. *Manuscript, Univ. of Maryland, http://www. cs. umd. edu/avik/projects/scandroidascaa*, 2(3), 2009.
102. Qi Gao, Fabian Abel, Geert-Jan Houben, and Yong Yu. A comparative study of usersï£¡ microblogging behavior on sina weibo and twitter. In *User modeling, adaptation, and personalization*, pages 88–101. Springer, 2012.
103. Yue Gao, Meng Wang, Zheng-Jun Zha, Jialie Shen, Xuelong Li, and Xindong Wu. Visual-textual joint relevance learning for tag-based social image search. *Image Processing, IEEE Transactions on*, 22(1):363–376, 2013.

104. Poonam Garg, CK Nagpal, and Swatie Bansal. Impact of random waypoint mobility model on hybrid routing protocols of scalable mobile ad hoc network. *International Journal of Innovative Research and Development*, 2(10), 2013.

105. Mark N Gasson, Eleni Kosta, Denis Royer, Martin Meints, and Kevin Warwick. Normality mining: Privacy implications of behavioral profiles drawn from gps enabled mobile phones. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 41(2):251–261, 2011.

106. Maíra Gatti, Paulo Cavalin, Samuel Barbosa Neto, Claudio Pinhanez, Cícero dos Santos, Daniel Gribel, and Ana Paula Appel. Large-scale multi-agent-based modeling and simulation of microblogging-based online social network. In *Multi-Agent-Based Simulation XIV*, pages 17–33. Springer, 2014.

107. G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K. Tan. Private queries in location based services: anonymizers are not necessary. In Jason Tsong-Li Wang, editor, *SIGMOD Conference*, pages 121–132. ACM, 2008.

108. G. Ghoshal, V. Zlatić, G. Caldarelli, and M. E. J. Newman. Random hypergraphs and their applications. *Physical Review E*, 79(6):066118, 2009.

109. M. Gjoka, M. Kurant, C.T. Butts, and A. Markopoulou. Walking in Facebook: A case study of unbiased sampling of OSNs. In *Proc. of the International Conference on Computer Communications (INFOCOM'10)*, pages 1–9, San Diego, CA, USA, 2010. IEEE.

110. Minas Gjoka, Maciej Kurant, Carter T Butts, and Athina Markopoulou. Walking in Facebook: A case study of unbiased sampling of OSNs. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

111. Derek Greene and Pádraig Cunningham. Producing a unified graph representation from multiple social network views. In *Proceedings of the 5th Annual ACM Web Science Conference*, pages 118–121. ACM, 2013.

112. M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42. ACM, 2003.

113. László Gyarmati and Tuan Anh Trinh. Measuring user behavior in online social networks. *Network, IEEE*, 24(5):26–31, 2010.

114. Hakan Hacigümüş, Bala Iyer, and Sharad Mehrotra. Ensuring the integrity of encrypted databases in the database-as-a-service model. In *Data and Applications Security XVII*, pages 61–74. Springer, 2004.

115. Eszter Hargittai and Eden Litt. The tweet smell of celebrity success: Explaining variation in twitter adoption among a diverse group of young adults. *New Media & Society*, 13(5):824–842, 2011.

116. C. Hawes, C.D. Phillips, M. Rose, S. Holan, and M. Sherman. A national survey of assisted living facilities. *Gerontologist*, 43(6):875–882, 2003.

117. Catherine Hawes, Charles D Phillips, Miriam Rose, et al. *High service or high privacy assisted living facilities, their residents and staff: Results from a national survey*. US Department of Health and Human Services Washington, DC, 2000.

118. Jorge L Hernandez-Ardieta, Ana I Gonzalez-Tablas, Jose M De Fuentes, and Benjamin Ramos. A taxonomy and survey of attacks on digital signatures. *computers & security*, 34:67–112, 2013.

119. B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Enhancing security and privacy in traffic-monitoring systems. *IEEE Pervasive Computing*, 5(4):38–46, 2006.

120. Jun Hong, Tao Wen, Quan Gu, and Gang Sheng. Query integrity verification based-on mac chain in cloud storage. In *Computer and Information Science (ICIS), 2014 IEEE/ACIS 13th International Conference on*, pages 125–129. IEEE, 2014.

121. Jeff Howe. The rise of crowdsourcing. *Wired magazine*, 14(6):1–4, 2006.

122. David John Hughes, Moss Rowe, Mark Batey, and Andrew Lee. A tale of two sites: Twitter vs. facebook and the personality predictors of social media usage. *Computers in Human Behavior*, 28(2):561–569, 2012.

123. T.D. Huynh, N. Jennings, and N. Shadbolt. Fire: An integrated trust and reputation model for open multi-agent systems. In *ECAI 2004: 16th European Conference on Artificial Intelligence, August 22-27, 2004, Valencia, Spain: Including Prestigious Applicants [sic] of Intelligent Systems (PAIS 2004): Proceedings*, volume 110, page 18. Ios PressInc, 2004.

124. T.D. Huynh, N.R. Jennings, and N.R. Shadbolt. Certified reputation: how an agent can trust a stranger. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1217–1224. ACM, 2006.

125. Trung Dong Huynh, Nicholas R Jennings, and Nigel R Shadbolt. Certified reputation: how an agent can trust a stranger. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 1217–1224. ACM, 2006.

126. Trung Dong Huynh, Nicholas R Jennings, and Nigel R Shadbolt. An integrated trust and reputation model for open multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 13(2):119–154, 2006.

127. Ren-Hung Hwang and Fu-Hui Huang. Socialcloaking: A distributed architecture for k-anonymity location privacy protection. In *Computing, Networking and Communications (ICNC), 2014 International Conference on*, pages 247–251. IEEE, 2014.

128. T. Iofciu, P. Fankhauser, F. Abel, and K. Bischoff. Identifying users across social tagging systems. In *Proc. of the International Conference on Weblogs and Social Media (ICWSM'11)*, Barcelona, Catalonia, Spain, 2011. The AAAI Press.

129. Paul Jaccard. *Etude comparative de la distribution florale dans une portion des Alpes et du Jura*. Impr. Corbaz, 1901.

130. Akshay Java, Xiaodan Song, Tim Finin, and Belle Tseng. Why we twitter: understanding microblogging usage and communities. In *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, pages 56–65. ACM, 2007.

131. Saeed Javanmardi, Mohammad Shojafar, Shahdad Shariatmadari, and Sima S Ahrabi. Fr trust: a fuzzy reputation–based model for trust management in semantic p2p grids. *International Journal of Grid and Utility Computing*, 6(1):57–66, 2015.

132. Long Jin, Yang Chen, Tianyi Wang, Pan Hui, and Athanasios V Vasilakos. Understanding user behavior in online social networks: A survey. *IEEE Communications Magazine*, 51(9):144–150, 2013.

133. A. Juels. Rfid security and privacy: A research survey. *Selected Areas in Communications, IEEE Journal on*, 24(2):381–394, 2006.

134. Ari Juels and Burton S Kaliski Jr. Pors: Proofs of retrievability for large files. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 584–597. Acm, 2007.

135. P. Kalnis, G. Ghinita, K. Mouratidis, and D. Papadias. Preventing location-based identity inference in anonymous spatial queries. *IEEE Trans. on Knowl. and Data Eng.*, 19(12):1719–1733, 2007.

136. H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. Random-data perturbation techniques and privacy-preserving data mining. *Knowl. Inf. Syst.*, 7(4):387–414, 2005.

137. G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: applications in VLSI domain. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 7(1):69–79, 1999.

138. Kate Keahey. Fine-grain authorization for resource management in the grid environment. In *Grid ComputingÃćÂĂÂŤGRID 2002*, pages 199–206. Springer, 2002.

139. S.M. Khan and K.W. Hamlen. Anonymouscloud: A data ownership privacy provider framework in cloud computing. 2012.

140. Yong-Ki Kim, Amina Hossain, Al-Amin Hossain, and Jae-Woo Chang. Hilbert-order based spatial cloaking algorithm in road network. *Concurrency and Computation: Practice and Experience*, 25(1):143–158, 2013.

141. Jon M Kleinberg. Navigation in a small world. *Nature*, 406(6798):845–845, 2000.

142. N. Korula and S. Lattanzi. An efficient reconciliation algorithm for social networks. In *Proc. of the International Conference on Very Large Data Bases (VLDB'14)*, pages 377–388, Hangzhou, Cina, 2014. VLDB Endowment.

143. Ahmed Kosba, Andrew Miller, Elaine Shi, Zikai Wen, and Charalampos Papamanthou. Hawk: The blockchain model of cryptography and privacy-preserving smart contracts. *University of Maryland and Cornell University*, 2015.

144. Kalliopi Kravari, Christos Malliarakis, and Nick Bassiliades. T-rex: A hybrid agent trust model based on witness reputation and personal experience. In *E-Commerce and Web Technologies*, pages 107–118. Springer, 2010.

145. Balachander Krishnamurthy. A measure of online social networks. In *Communication Systems and Networks and Workshops, 2009. COMSNETS 2009. First International*, pages 1–10. IEEE, 2009.

146. J. Krumm. Inference attacks on location tracks. In *International Conference Pervasive on Pervasive Computing*, Lecture Notes in Computer Science, pages 127–143. Springer, 2007.

147. Wei-Shinn Ku, Ling Hu, Cyrus Shahabi, and Haixun Wang. A query integrity assurance scheme for accessing outsourced spatial databases. *Geoinformatica*, 17(1):97–124, 2013.

148. Hendrik Kuijs, Carina Rosencrantz, and Christoph Reich. A context-aware, intelligent and flexible ambient assisted living platform architecture. *Cloud Computing*, 2015.

149. Mariantonietta La Polla, Fabio Martinelli, and Daniele Sgandurra. A survey on security for mobile devices. *Communications Surveys & Tutorials, IEEE*, 15(1):446–471, 2013.

150. Risto Laurikainen. Secure and anonymous communication in the cloud. *Aalto University School of Science and Technology, Department of Computer Science and Engineering, Tech. Rep. TKK-CSE-B10*, 2010.

151. Gianluca Lax, Francesco Buccafurri, and Gianluca Caminiti. Digital document signing: Vulnerabilities and solutions. *Information Security Journal: A Global Perspective*, 2015.

152. Byoungcheon Lee and Kwangjo Kim. Fair exchange of digital signatures using conditional signature. In *Symposium on Cryptography and Information Security*, pages 179–184, 2002.

153. R. T. Leenders. Modeling social influence through network autocorrelation: constructing the weight matrix. *Social Networks*, 24(1):21–47, 2002.

154. EA Leicht, Petter Holme, and Mark EJ Newman. Vertex similarity in networks. *Physical Review E*, 73(2):026120, 2006.

155. J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. *The Journal of Machine Learning Research*, 11:985–1042, 2010.

156. Jure Leskovec, Lada A Adamic, and Bernardo A Huberman. The dynamics of viral marketing. *ACM Transactions on the Web (TWEB)*, 1(1):5, 2007.

157. Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, and Jeffrey S Rosenschein. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 919–927. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

158. Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *International Conference on Financial Cryptography and Data Security*, pages 528–547. Springer, 2015.

159. Feifei Li, Marios Hadjieleftheriou, George Kollios, and Leonid Reyzin. Dynamic authenticated index structures for outsourced databases. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 121–132. ACM, 2006.

160. Feifei Li, Marios Hadjieleftheriou, George Kollios, and Leonid Reyzin. Authenticated index structures for aggregation queries. *ACM Transactions on Information and System Security (TISSEC)*, 13(4):32, 2010.

161. Jin Li, Xiaofeng Chen, Jingwei Li, Chunfu Jia, Jianfeng Ma, and Wenjing Lou. Fine-grained access control system based on outsourced attribute-based encryption. In *Computer Security–ESORICS 2013*, pages 592–609. Springer, 2013.

162. Seung-Hwan Lim, Sang-Wook Kim, Sunju Park, and Joon Ho Lee. Determining content power users in a blog network: an approach and its applications. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 41(5):853–862, 2011.

163. X. Lin, R. Lu, D. Kwan, and X.S. Shen. REACT: An RFID-based privacy-preserving children tracking scheme for large amusement parks. *Computer Networks*, 54(15):2744–2755, 2010.

164. L. Lovász. Random walks on graphs: A survey. *Combinatorics, Paul Erdos is Eighty*, 2(1):1–46, 1993.

165. Adolfo Lozano-Tello and Asunción Gómez-Pérez. Ontometric: A method to choose the appropriate ontology. *Journal of Database Management*, 2(15):1–18, 2004.

166. L. Lü and T. Zhou. Link Prediction in Complex Networks: A Survey. *Physica A: Statistical Mechanics and its Applications*, 390(6):1150–1170, 2011.

167. Marcelo Maia, Jussara Almeida, and Virgílio Almeida. Identifying user behavior in online social networks. In *Proceedings of the 1st workshop on Social network systems*, pages 1–6. ACM, 2008.

168. Bradley A Malin, Khaled El Emam, and Christine M O'Keefe. Biomedical data privacy: problems, perspectives, and recent advances. *Journal of the American Medical Informatics Association*, 20(1):2–6, 2013.

169. Lukas Malina and Jan Hajny. Efficient security solution for privacy-preserving cloud services. In *Telecommunications and Signal Processing (TSP), 2013 36th International Conference on*, pages 23–27. IEEE, 2013.

170. Tsutomu Matsumoto. Human–computer cryptography: An attempt. *Journal of Computer Security*, 6(3):129–149, 1998.

171. Ashish K Maurya, Dinesh Singh, Ajeet Kumar, and Ritesh Maurya. Random waypoint mobility model based performance estimation of on-demand routing protocols in manet for cbr applications. In *Computing for Sustainable Global Development (INDIACom), 2014 International Conference on*, pages 835–839. IEEE, 2014.

172. Roy Maxion, Kymie Tan, et al. Benchmarking anomaly-based detection systems. In *Dependable Systems and Networks, 2000. DSN 2000. Proceedings International Conference on*, pages 623–630. IEEE, 2000.

173. A.J. Menezes, P.C. Van Oorschot, and S.A. Vanstone. *Handbook of applied cryptography*. CRC, 1997.

174. Ralph C Merkle. A certified digital signature. In *Advances in CryptologyâĂŤCRYPTOâĂŹ89 Proceedings*, pages 218–238. Springer, 1989.

175. P. Mika. Ontologies are us: A unified model of social networks and semantics. In *The Semantic Web–ISWC 2005*, pages 522–536. Springer, 2005.

176. Miranda Mowbray and Siani Pearson. A client-based privacy manager for cloud computing. In *Proceedings of the fourth international ICST conference on COMmunication system softWAre and middlewaRE*, page 5. ACM, 2009.

177. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

178. Moni Naor and Benny Pinkas. Visual authentication and identification. In *Advances in Cryptologyï£¡CRYPTO'97*, pages 322–336. Springer, 1997.

179. Maithili Narasimha and Gene Tsudik. Dsac: integrity for outsourced databases with signature aggregation and chaining. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 235–236. ACM, 2005.

180. A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *Proc. of the International IEEE Symposium on Security and Privacy*, pages 173–187, Oakland, California, USA, 2009. IEEE Computer Society.

181. Mohammad Nauman, Sohail Khan, and Xinwen Zhang. Apex: extending android permission model and enforcement with user-defined runtime constraints. In *Proceedings*

of the 5th ACM Symposium on Information, Computer and Communications Security, pages 328–332. ACM, 2010.

182. M. E. J. Newman, D. J. Watts, and S. H. Strogatz. Random graph models of social networks. *Proceedings of the National Academy of Sciences*, 99(suppl 1):2566–2572, 2002.
183. Mark EJ Newman. Assortative mixing in networks. *Physical review letters*, 89(20):208701, 2002.
184. Muhammad Saqib Niaz and Gunter Saake. Merkle hash tree based techniques for data integrity of outsourced data. In *GvD*, pages 66–71, 2015.
185. Ben Niu, Qinghua Li, Xiaoyan Zhu, Guohong Cao, and Hui Li. Achieving k-anonymity in privacy-aware location-based services. In *Proc. IEEE INFOCOM*, 2014.
186. S. Noor and K. Martinez. Using social data as context for making recommendations: an ontology based approach. In *Proceedings of the 1st Workshop on Context, Information and Ontologies*, page 7. ACM, 2009.
187. Jane E Nordholt, Richard John Hughes, Jane Marie Riese, Christine Marie Ahrens, Charles Glen Peterson, and James William Harrington. Scalable software architecture for quantum cryptographic key management, August 16 2013. US Patent App. 14/423,551.
188. Machigar Ongtang, Stephen McLaughlin, William Enck, and Patrick McDaniel. Semantically rich application-centric security in android. *Security and Communication Networks*, 5(6):658–673, 2012.
189. Y. Ouyang, Y. Xu, Z. Le, G. Chen, and F. Makedon. Providing location privacy in assisted living environments. In *PETRA '08: Proceedings of the 1st international conference on PErvasive Technologies Related to Assistive Environments*, pages 1–8, New York, NY, USA, 2008. ACM.
190. Elliot T Panek, Yioryos Nardis, and Sara Konrath. Mirror or megaphone?: How relationships between narcissism and social networking site use differ on facebook and twitter. *Computers in Human Behavior*, 29(5):2004–2012, 2013.
191. HweeHwa Pang, Arpit Jain, Krithi Ramamritham, and Kian-Lee Tan. Verifying completeness of relational query results in data publishing. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 407–418. ACM, 2005.
192. George Papadakis, Konstantinos Tserpes, Emmanuel Sardis, Magdalini Kardara, Athanasios Papaoikonomou, and Fotis Aisopos. Social media meta-api: leveraging the content of social networks. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 271–274. ACM, 2012.
193. Tiffany A Pempek, Yevdokiya A Yermolayeva, and Sandra L Calvert. College students' social networking experiences on facebook. *Journal of Applied Developmental Psychology*, 30(3):227–238, 2009.
194. Min Peng, ZhengQuan Xu, ShaoMing Pan, Rui Li, and Tengyue Mao. Agenttms: A mas trust model based on agent social relationship. *Journal of computers*, 7(6):1535–1542, 2012.
195. Joseph Poon and Thaddeus Dryja. The bitcoin lightning network, 2015.

196. Robert Porzel and Rainer Malaka. A task-based approach for ontology evaluation. In *ECAI Workshop on Ontology Learning and Population, Valencia, Spain*. Citeseer, 2004.

197. Tal Rabin. Robust sharing of secrets when the dealer is honest or cheating. *Journal of the ACM (JACM)*, 41(6):1089–1109, 1994.

198. Parisa Rashidi and Alex Mihailidis. A survey on ambient-assisted living tools for older adults. *IEEE journal of biomedical and health informatics*, 17(3):579–590, 2013.

199. Erzsébet Ravasz, Anna Lisa Somera, Dale A Mongru, Zoltán N Oltvai, and A-L Barabási. Hierarchical organization of modularity in metabolic networks. *science*, 297(5586):1551–1555, 2002.

200. Mohamed Amine Riahla, Karim Tamine, and Philippe Gaborit. A protocol for file sharing, anonymous and confidential, adapted to p2p networks. In *Sciences of Electronics, Technologies of Information and Telecommunications (SETIT), 2012 6th International Conference on*, pages 549–557. IEEE, 2012.

201. R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, 1978.

202. Pablo Rodriguez. Web Infrastructure for the 21st Century. In *18th International World Wide Web Conference*, April 2009.

203. Larry D Rosen, K Whaling, S Rab, L Mark Carrier, and Nancy A Cheever. Is facebook creating disorders? the link between clinical symptoms of psychiatric disorders and technology use, attitudes and anxiety. *Computers in Human Behavior*, 29(3):1243–1254, 2013.

204. Brian Lee Yung Rowe. Using social network graph analysis for interest detection. *arXiv preprint arXiv:1410.0316*, 2014.

205. Sushmita Ruj, Milos Stojmenovic, and Amiya Nayak. Decentralized access control with anonymous authentication of data stored in clouds. *Parallel and Distributed Systems, IEEE Transactions on*, 25(2):384–394, 2014.

206. Jordi Sabater and Carles Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 475–482. ACM, 2002.

207. Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

208. Pierangela Samarati. Protecting respondents identities in microdata release. *Knowledge and Data Engineering, IEEE Transactions on*, 13(6):1010–1027, 2001.

209. Pierangela Samarati. Data security and privacy in the cloud. In *ISPEC*, pages 28–41. Springer, 2014.

210. Joshua Schiffman, Xinwen Zhang, and Simon Gibbs. Dauth: Fine-grained authorization delegation for distributed web application consumers. In *Policies for Distributed Systems and Networks (POLICY), 2010 IEEE International Symposium on*, pages 95–102. IEEE, 2010.

211. Fred B Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys (CSUR)*, 22(4):299–319, 1990.

212. Aamir Shahzad, Malrey Lee, Young-Keun Lee, Suntae Kim, Naixue Xiong, Jae-Young Choi, and Younghwa Cho. Real time modbus transmissions and cryptography security

designs and enhancements of protocol sensitive information. *Symmetry*, 7(3):1176–1210, 2015.

213. Mohamed Shehab, Said Marouf, and Christopher Hudel. Roauth: recommendation based open authorization. In *Proceedings of the Seventh Symposium on Usable Privacy and Security*, page 11. ACM, 2011.

214. Jianqiang Shen, Oliver Brdiczka, and Yiye Ruan. A comparison study of user behavior on facebook and gmail. *Computers in Human Behavior*, 29(6):2650–2655, 2013.

215. Lal Vikram Singh, Amol V Bole, and Shailesh Kumar Yadav. Security issues of cloud computing-a survey. 2015.

216. Yonatan Sompolinsky and Aviv Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. *IACR Cryptology ePrint Archive*, 2013:881, 2013.

217. Joo-Han Song, Vincent W. S. Wong, and Victor C. M. Leung. Wireless location privacy protection in vehicular ad-hoc networks. *Mobile Networks and Applications*, 15:160 – 171, 2010.

218. Thorvald Sørensen. {A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons}. *Biol. Skr.*, 5:1–34, 1948.

219. L. Specia and E. Motta. Integrating folksonomies with the semantic web. In *The semantic web: research and applications*, pages 624–639. Springer, 2007.

220. Jessica Staddon, David Huffaker, Larkin Brown, and Aaron Sedley. Are privacy concerns a turn-off?: engagement and privacy in social networks. In *Proceedings of the eighth symposium on usable privacy and security*, page 10. ACM, 2012.

221. Brian Stanford-Smith and Paul T Kidd. *E-business: key issues, applications and technologies*. IOS press, 2000.

222. Statista. The statistics portal. `http://www.statista.com/statistics`, 2015.

223. A. Stewart, E. Diaz-Aviles, W. Nejdl, L. B. Marinho, A. Nanopoulos, and L. Schmidt-Thieme. Cross-tagging for personalized open social networking. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 271–278. ACM, 2009.

224. A. Stubblefield, J. Ioannidis, and A. D. Rubin. A key recovery attack on the 802.11b wired equivalent privacy protocol (wep). *ACM Trans. Inf. Syst. Secur.*, 7(2):319–332, 2004.

225. D. Stutzback, R. Rejaie, N. Duffield, S. Sen, and W. Willinger. On unbiased sampling for unstructured peer-to-peer networks. In *Proc. of the International Conference on Internet Measurements*, pages 27–40, Rio De Janeiro, Brasil, 2006. ACM.

226. Shital S Suryawanshi and Vinod S Wadne. Securing health care data in collaborative data publishing using mapreduce framework. 2015.

227. WT Luke Teacy, Jigar Patel, Nicholas R Jennings, and Michael Luck. Travos: Trust and reputation in the context of inaccurate information sources. *Autonomous Agents and Multi-Agent Systems*, 12(2):183–198, 2006.

228. Jaime Teevan, Daniel Ramage, and Merredith Ringel Morris. # twittersearch: a comparison of microblog search and web search. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 35–44. ACM, 2011.

229. TheGuardian. Italy fines TripAdvisor €500,000 over false reviews. `http://www.theguardian.com/travel/2014/dec/23/ italy-fines-tripadvisor-500000`, 2014.

230. R. Trujillo-Rasua and J. Domingo-Ferrer. On the privacy offered by *(k, δ)*-anonymity. *Information Systems*, 38(4):491–494, 2013.

231. Rolando Trujillo-Rasua and Agusti Solanas. Efficient probabilistic communication protocol for the private identification of rfid tags by means of collaborative readers. *Computer Networks*, 55(15):3211–3223, 2011.

232. Sebastián Valenzuela, Namsu Park, and Kerk F Kee. Is there social capital in a social network site?: Facebook use and college students' life satisfaction, trust, and participation1. *Journal of Computer-Mediated Communication*, 14(4):875–901, 2009.

233. Katrien Van Cleemput. Friendship type, clique formation and the everyday use of communication technologies in a peer group: A social network analysis. *Information, Communication & Society*, 15(8):1258–1277, 2012.

234. A. Vilamovska, E. Hatziandreu, H. R. Schindler, C. Van Oranje-Nassau, H. De Vries, and J. Krapels. Study on the requirements and options for rfid application in healthcare. 2009.

235. J. Vosecky, D. Hong, and V.Y. Shen. User identification across multiple social networks. In *Proc. of the International Conference on Networked Digital Technologies (NDT'09)*, pages 360–365, Ostrava, the Czech Republic, 2009. IEEE Press.

236. Cong Wang, Qian Wang, Kui Ren, Ning Cao, and Wenjing Lou. Toward secure and dependable storage services in cloud computing. *Services Computing, IEEE Transactions on*, 5(2):220–232, 2012.

237. Cong Wang, Qian Wang, Kui Ren, and Wenjing Lou. Privacy-preserving public auditing for data storage security in cloud computing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. Ieee, 2010.

238. Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li. Enabling public auditability and data dynamics for storage security in cloud computing. *Parallel and Distributed Systems, IEEE Transactions on*, 22(5):847–859, 2011.

239. X. Wang, F. Wei, X. Liu, M. Zhou, and M. Zhang. Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1031–1040. ACM, 2011.

240. Yongjin Wang and Konstantinos N Plataniotis. An analysis of random projection for changeable and privacy-preserving biometric verification. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 40(5):1280–1293, 2010.

241. Duncan J Watts, Jonah Peretti, and Michael Frumin. *Viral marketing for the real world*. Harvard Business School Pub., 2007.

242. Debra Aho Williamson. Social network marketing: Ad spending and usage. *Social Network Marketing, Report by Debra Aho Williamson*, 2007.

243. Christo Wilson, Bryce Boe, Alessandra Sala, Krishna PN Puttaswamy, and Ben Y Zhao. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 205–218. Acm, 2009.

244. Min Xie, Haixun Wang, Jian Yin, and Xiaofeng Meng. Integrity auditing of outsourced data. In *Proceedings of the 33rd international conference on Very large data bases*, pages 782–793. VLDB Endowment, 2007.

245. Zhiyong Xu, Rui Min, and Yiming Hu. Hieras: a dht based hierarchical p2p routing algorithm. In *Parallel Processing, 2003. Proceedings. 2003 International Conference on*, pages 187–194. IEEE, 2003.

246. Ziwei Yang, Shen Gao, Jianliang Xu, and Byron Choi. Authentication of range query results in mapreduce environments. In *Proceedings of the third international workshop on Cloud data management*, pages 25–32. ACM, 2011.

247. S. Ye, J. Lang, and F. Wu. Crawling online social graphs. In *Proc. of the International Asia-Pacific Web Conference (APWeb'10)*, pages 236–242, Busan, Korea, 2010. IEEE.

248. R. Zafarani and H. Liu. Connecting corresponding identities across communities. In *Proc. of the International Conference on Weblogs and Social Media (ICWSM'09)*, San Jose, CA, USA, 2009. The AAAI Press.

249. Degan Zhang, Xiang Wang, Xiaodong Song, and Dexin Zhao. A novel approach to mapped correlation of id for rfid anti-collision. *Services Computing, IEEE Transactions on*, 7(4):741–748, 2014.

250. Z. Zhang and C. Liu. A hypergraph model of social tagging networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2010(10):P10005, 2010.

251. Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. Comparing twitter and traditional media using topic models. In *Advances in Information Retrieval*, pages 338–349. Springer, 2011.

252. Bu Zhong, Marie Hardin, and Tao Sun. Less effortful thinking leads to more social networking? the associations between the use of social network sites and personality traits. *Computers in Human Behavior*, 27(3):1265–1271, 2011.

253. Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. Predicting missing links via local information. *The European Physical Journal B*, 71(4):623–630, 2009.

254. Sheryl Zimmerman. *Assisted living: Needs, practices, and policies in residential care for the elderly*. JHU Press, 2001.

255. Lili Nemec Zlatolas, Tatjana Welzer, Marjan Heričko, and Marko Hölbl. Privacy antecedents for sns self-disclosure: The case of facebook. *Computers in Human Behavior*, 45:158–167, 2015.

256. A. Zwierko and Z. Kotulski. A light-weight e-voting system with distributed trust. *Electronic Notes in Theoretical Computer Science*, 168:109–126, 2007.