UNIVERSITÀ DEGLI STUDI DELL'INSUBRIA
Dipartimento di Scienze Teoriche e Applicate - DISTA

Dissertation

COLLABORATIVE EXECUTION OF SECURITY-RELATED PROCESS

Philosophy of Doctor
In
Computer Science

PhD Candidate: Ngoc Hong Tran
Advisors: Prof. Elena Ferrari, Prof. Barbara Carminati

Italy, 11th January 2016

I would like to dedicate this thesis to my beloved parents,
Hue Hong Tran and Sam Thi Nguyen,
and my little nephews, Anh Minh Tran and Lam Bao Tran.

Acknowledgements

I would like to express my deepest gratitude to my beloved parents, Hue and Sam, for carrying me to this life, bringing carefully me up, and educating me thoughtfully. They always support me silently behind, give me wings to fly farther in my career and a root to come back whenever I am tired. I would like to thank my sister, Hanh Ngoc Tran, and my brother, Huy Ngoc Tran, for their advice in life and for sharpening my spirit when I have been living apart from my family.

I cannot obtain this Ph.D. without my excellent advisors, professor Elena Ferrari and professor Barbara Carminati. I would like to express the most sincere gratitude to my advisors. They played the key roles in my Ph.D. In addition to their academic guidance, they have shown a great deal of patience towards me and supported my research in a way that goes beyond a professional relationship. Their caring attitude has always given me the strength to continue my research endeavor.

I would like to thank Leila Bahri and Natasha Poposka for always standing by my side and being my best friends during years in here. I would like to thank my friends Shuai Li and Naime Laleh for sharing the office and spending some time with me. I would like to thank all lab colleagues, Dr. Marco Tarini, Dr. Pietro Colombo, Dr. Valentina Pedoia, Dr. Giulio Liu, Dr. Cuneyt Arkora, Dr. Lorenzo Bossi, Dr. Michele Giuglielmi, and all my Italian friends in here, for showing me Italian hospitality and enriching my life greatly. Without them, my Ph.D. would never be such a great experience.

I would like to thank Mauro Santabarbara for his time in helping me with the computer that I used during my research. I would like to thank Roberta Viola for aiding me in processing the paperwork during my Ph.D.

Finally, I would like to thank the Italian people for their warmth and grace, which made me feel at home during these years.

Abstract

So far the number of users on the social network sites has been increasing year-on-year. In the meanwhile, the price of mobile devices drops vertically. Both events push up vigorously the quantity of social users who use mobile devices. This results in a need for collaboration among mobile social users to attain the common goals. The fact that the huge number of users goes online lifts forcefully up the number of services over the Internet. The user requirements for services then get more complex. As a consequence, responding such a complex service request needs a composition of several single services.

However, the collaboration among several participants can make the personal data disclosed to the other sides during the collaborative process. Despite the fact that all participants comply with the common protocol and agree on contributing their personal data, they do not trust adequately in each other. This leads to the requirement of preserving user privacy and securing the user data against the other sides.

In this dissertation, we investigate different collaboration and participant types. More specifically, they are centralized and decentralized models among users/services. We also concern the communication environment, e.g., mobile network, Internet, or mobile ad-hoc network. The protocols that are enforced in a mobile network environment have more rigorous constraints due to limited physical and performance resources. For preserving user privacy and data security, we study and select different rational and effective cryptography algorithms then apply them into the collaborative protocols. For more understanding, we propose several collaboration scenarios and present them in detail in the upcoming chapters. Each scenario describes a combination of different collaborative type, participant type and network environment. For each scenario, we address issues on privacy preserving and data security and propose selective solutions. In addition, we propose methods for improving network performance. The experimental results show that our proposals are efficient and effective.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Background and Motivation

In sociology, social network had been studied since 1978 [11] [134]. The social network has evolved through remarkable milestones, and debuted with Bulletin Board System (BBS) carrying users the means for exchanging the data using phone lines. Though BBS had many limits, it set up the initial interactive environment for users. A real social network site, named Geocities, was first exhibited in 1994. Initiatives following Geocities are respectively: AOL (in 1995), Friendster (in 2002), MySpace (in 2004), Youtube (in 2005), Tweeter (in 2006), Pinterest and Instagram (in 2010). An amazing explosion of the social network site happened in 2008 when Facebook opened for external users. Facebook had been tested only with Havard University students for four years, then overtook MySpace to become a leader of social network sites. Since then, a new page for the age of social network sites has been turned. The social network sites continue to increase year-on-year in the number of users. Emarketer predicts that the development rate of 2016 will hit 8.9% and that 2.55 billion people worldwide will use social networks by 2016 [49].

With the tremendous growth of social network sites, users have numerous convenient means to keep in touch and share every moment of the life with the rest of the world. Social users start integrating their daily routine to the social network. When people get used to socialize online, the requirement for social collaboration rises up. Plenty of invented platforms supporting users in collaboration have lately come up. For an example, Acrobat.com allows teams to collaborate on documents by their browsers, Socialcast helps employees discuss projects remotely through a micro-blogging service accessible from smart phones, etc. [145]. Such a quickly rising number of social network users turn all of social network sites into a social media platform. With benefits from social media, the commercial services grow up swiftly, user requirements on products/services are more and more complex. This leads

to the needs of collaborating several commercial services in social network as well. As a result, the collaborations are then more focused [107]. Inspired by this socially collaborative requirement, collaborations among several participants are considered as an essential need.

However, not all participants in the collaboration know each other. This reduces significantly the trustworthiness among them, as the collaboration among them possibly reveals their personal data to the other sides. Even though all participants comply with the common protocol and agree on contributing their personal data, they do not trust adequately in each other. As a result, the requirements of preserving user privacy and securing the user data against the other sides are a substantial need. This dissertation addresses the user privacy and data security related issues and discusses the different solutions. Then, we select the effective and efficient ones applied to the collaboration protocols.

As mentioned above, in the social media the collaboration is deployed not only for users but also for organizations, i.e., service providers. In this dissertation, both types of collaborator (i.e., user and service provider) are considered in detail through the proposed multiparty collaborative protocols. Hereafter, the term "user" does not imply only the individual who participates in the social media, but also the electronic device, relating to that individual, where local data is stored and which has the computing capability. As well, service provider does not only imply the organization, but also the service supplied to users and hosted on servers.

As pointed out in the technical note of IBM [129], collaboration, is organized according to the network models, i.e., decentralization or centralization. Let us give a reminder of these two network models. According to the centralized model, there exists a central controller that is requested to be powerful so as to be able to work with multiple partners simultaneously. It is also the one that coordinates all partners and synchronizes processes in the system. A large bandwidth is one key to this model. The single point failure is a shortcoming. However, it is easy to deploy this model. Therefore, the centralized model has been investigated carefully, and continues to be researched for commercial products or services, e.g., Huawei technology services are built up and developed in the centralized model [30].

To overcome the limits of the centralized model, a decentralized model is one choice. This model lacks the central controller, thus the peers in this model will do the central controller's job beside performing their own tasks. In particular, they independently search a peer able to cooperate with them and then invoke the selected. It is harder to implement this model, compared with the centralized model. In a meanwhile, considered as the most important technological trend [141][123], decentralized model is a very promising approach, for an example, IBM have agreed on the future trend of decentralized model and run a famous project, named ADEPT, according to the decentralized model [139]. Both of collaboration

models are today used in collaborative systems. No model can be replaced by the other, as every of them has advantages and disadvantages, and both need to be investigated for being systematically and rationally applied.

Mobile social networks. Along with the expanse of social collaboration, the social media goes mobile as well, thanks to the revolution of mobile devices. Since electronic sensors get much cheaper, the price of a mobile device drops drastically. This brings users a chance to own a mobile device easily. Gartner predicted that by 2020, 75 percent of smartphone buyers will pay less than $100 for a device [48]. Since all users have mobile devices do the jobs of Internet and even television, mobile device overtakes the personal computers, and mobile network can replace Internet [17]. The mobile devices turn out to be a vital communication means attaching to the daily routine of people. According to Gartner, more than 50 percent of users are predicted to use a tablet or smartphone first for all online activities by 2018; and according to eMarketer, the total mobile phone users are likely to reach 5.13 billion users globally by 2017 [133]. Under the influence of mobile devices and social collaboration on the worldwide market, the combination of them, mobile social collaboration, becomes the most popular.

Motivated by the above discussion, the dissertation has focused on the collaborations among mobile users/devices and exploited the connections among them in social network sites.

1.1.1 Secure-aware Mobile social collaboration

According to Statista, by 2018, over 75 percent of Facebook users worldwide will access it through their mobile phone [135]. Social users' daily activities are now done in mobile social network, such as, contacting friends, entertaining, etc. as well as activities corresponding to monetary transactions, such as, online shopping, etc. As a result, the online social payment comes up for the online shopping through social network sites [63]. This is a sort of online payment between one user and one company/provider. Beside that, new digital methods of making person-to-person (P2P) payments also get a rise. This allows customers to use their mobile phone to support mobile Person-to-Person payments. Gartner [47] forecasts that in 2016 there will be 448 million mobile payment users in a market worth 617 billion USD, and the global mobile transaction volume and value averages 42% annual growth between 2011 and 2016. Despite the success of these new digital P2P payment methods, we believe that, to fully enable this increasing rise of digital wallets, relevant challenges still need to be addressed, particularly over social media. However, as it happens in the real life when we tend to lend somebody money, we make decisions based on what we know about him/her. This cannot be applied in social network since there are people we do not know personally.

The only thing which can help us evaluate a user's trustworthiness is to exploit the social connections, between them and us, for quantifying the amount of trust we have for them, then judge their reputation by the quantified trust. This performance is called trust computing. As a step in the direction combining social collaboration and online social person-to-person payment, this dissertation leverages social network connections to support social users in computing the trust of a person who wants to borrow them money before deciding a money transfer [22, 23, 25].

The fact that computing the trust of a social user is discovering the path connecting two social users. This requires users on the path need to collaborate and contribute their relationship information to trust computing. However, there is a privacy requirement that users do not want the others to read their own data, e.g., fullname, credit card, etc., and they are not also allowed to read the others' data, even the output data from the path discovery process. For this purpose, there is a need to investigate cryptography algorithms and to apply the suitable ones into the proposed collaborative protocols. Selecting the cryptography algorithms depends on their properties. For an example, the encryption algorithm RSA can be used for the wired network (i.e., Ethernet) since the bandwidth is larger, but it runs slowly in the wireless network with mobile devices. Beside, there is another issue that a user can identify his/her direct contacts, but has a requirement on preserving his/her identity against contacts connecting indirectly to him/her. As a result, security and privacy issues in collaborative execution of those security related processes need to be addressed. Particularly, we emphasize to preserve the communications confidentiality, the authenticity and trustworthiness of communication partners, message integrity, and user privacy, and give possible solutions in the dissertation.

Moreover, parties in the collaborative process communicate together by transferring their messages through the network. The a naive messages propagation, the broadcast technique is used. It implies that a user forwards the received message to all of its contacts. This increases exponentially the number of out-going messages, and slows dramatically down the network bandwidth. Beside the propagation, another factors influencing on the network performance, the message size and the number of out-going messages. The key size of adopted cryptography algorithms can lengthen the message load, causing a deep impact on the bandwidth. As a result, the necessity of revising the generated messages and the propagation method optimization should be considered thoughtfully. The dissertation presents some methods of optimizing the generated message capacity as well as the number of out-going messages. Preliminary results on this have been published in [22, 23, 25].

Let us mention a missing crucial factor, that is, the specific communication means in which the collaborative protocol can operate well. We target Mobile Ad-hoc NETwork

(MANET) as a rational communication means. MANET has been investigated since the 1990's and recent predictions about the future research trends in MANET determined that it has potentials to be continuously developed [85], [126]. Since MANET's properties fit the realistic requirements of many scenarios, such as availability, cost saving, self-organized and infrastructure-less architecture, the applicability of MANET is extremely large. For instance, MANET applications have been developed on tactical networks, emergency, as well as education, context aware services, entertainment, military services [62]. Additionally, several applications have been deployed over MANET. For instance, available locations of users on their mobile devices have been exploited, among others, by Foursquare[1] and Gowalla[2], which are location-based social networking services for mobile devices, and by *Last.fm Festival*[3] that suggests a list of music festivals to users near the event locations. As further relevant examples, *TerraNet*[4] supports mobile phone calls without a connection through a cell tower; Mobile Chedar [7] is a middleware prototype with a mobile peer-to-peer learning environment application using Bluetooth; AdSocial [127] is a software platform supporting social network applications in ad hoc networks targeting small-scale scenarios, such as friends playing a game on the train or co-workers sharing calendar information, as well as, conference participants establishing voice-video calls, chat, or play games. In addition, we can cite MobiClique [116], a mobile social software, that allows people to maintain and to extend their online social networks through opportunistic connections between neighboring devices, and What's Up [97] an application providing spontaneous social networks in ad hoc networks, such as conferences and expositions. As witnessed by the above mentioned services, plenty of MANET applications have been deployed successfully, and particularly for scenarios where local networks with a high density of users are available (e.g., a conference room, library, supermarket, stadium, park, university campus, company buildings). This typical MANET scenario fits well the one in which P2P mobile payment applications can be used. Hence, we strongly believe that there is a need of deploying P2P payments over MANET. For this reason, the dissertation investigates how to deploy the mobile P2P payment over MANET. To our knowledge, this work is the first one exploiting social network relationships for P2P payments over MANET. Furthermore, as P2P payment collaborative protocol is performed over MANET, another crucial issue is spotted in case connections in social graph and MANET are not the same. The collaborative protocol needs to synchronize messages moving through users in social graph and nodes in MANET. Another minor but important issue is that MANET nodes are dynamic and change the network structure by partitioning it

[1] https://foursquare.com/about
[2] http://gowalla.com/
[3] http://www.last.fm/festivals
[4] http://terranet.se/

into several sub-networks. There is a need for connecting those sub-networks and enabling them to transfer data to each other smoothly. The solutions of these issues are going to be presented in the upcoming part of the dissertation [25].

1.1.2 Web based collaboration

Web 2.0 technology comes in 2004 and gets a powerful tool for implementing and helping social network grow up [31]. When the number of user accessing services by web browsers increased steeply, in order to comfort users in accessing services, the web 2.0 based services have been developed quickly by providers. Since then, user requirements on services get more and more complex, which leaded to the advent of collaborations among web service providers. inspired with above discussion, this dissertation also investigates and goes deeper collaborations among web service providers.

Let us recall quickly the definition and functionality of a web service. A web service is a software system designed to support interoperable application-to-application interactions over the Internet. One of the major goals of web services is to make easier their composition to form more complex services. Two are the main techniques for web service composition, namely, orchestration [33, 106, 65, 75] and choreography [5, 149, 73]. Choreography is a decentralized model with a dynamic sequence of web services used for B2B (Business-to-Business) applications. In contrast, orchestration has a BPEL (Business Process Execution Language) engine [110] as a central mediator to control a static sequence of activities described in a BPEL-based file. In this dissertation, we focus on both composition models to study how crucially they influence on collaborations based on centralized model [24] and decentralized model [21].

1.1.3 Secure-aware web composition

Web service orchestration. With the web service orchestration, we assume the workflow underlying the business process is encoded into a BPEL document and processed by a server hosting a BPEL engine (hereafter, the *broker*). According to the orchestration paradigm, the broker coordinates the invocation of services involved in the composition, i.e., partner services, by passing the needed parameters. In general, all previous proposals for the service orchestration model consider the broker as a trusted entity (See for example [138, 155, 109]). In these works, data from users or services come to the broker, the broker reads and processes them in a defined protocol. The data and user privacy can be preserved by matching policy at the broker side upon the assumption of the trusted broker. As such, they never pay attention to the fact that the broker is able to access several pieces of sensitive data, such as: the data

given as input by the user invoking the composite service, the final outcome of the composite service, as well as internal parameters (i.e., variables and values generated as output by partner services). The same issue can be caused with web service partners. We believe there is a need to protect them against improper access and usage from partner services as well as the broker. In this dissertation, we select the suitable encryption algorithms based on which we propose a secure protocol against the untrusted broker and web service partners [24].

Web service choreography. Recently literature shows that there is an increasing interests in decentralized composition as the orchestration paradigm suffers of single-point failure problem as well as requires powerful and reliable brokers. According to the choreography paradigm, once the business process and the workflow behind the web service composition has been defined, the composite service is deployed by invoking the service that has to perform the first activity. When this service correctly terminates the activity, it searches a service able to carry on the next activity in the workflow. This is then directly invoked by the service that has just terminated, without the presence of a broker (like in the orchestration model). It is relevant to note that regardless of the adopted paradigm, a crucial task is the selection of the service to be assigned to each activity in the workflow. As a matter of fact, service compositions can be driven by constraints on service selection. As discussed in [131], these requirements can be posed both by users requiring the composed service as well as by the companies provisioning the atomic services. In general, requirements can refer to several dimensions, such as Quality of Service (QoS) [86] or security [24], as services might have strong security requirements on services with which they have to cooperate during the service composition deployment. However, to the best of our knowledge none of them considers that the evaluation of these criteria requires to expose private information of users and providers, regardless of whom poses the requirements (i.e., users or providers) and the nature of these requirements (e.g., QoS, security, etc.). The only paper we are aware of dealing with these issues is [132], where a centralized and privacy aware approach for service selection of composite services has been proposed. A prominent point is that the peer-to-peer nature of the choreography model imposes to revise the way requirements have to be enforced. Indeed, the absences of a broker entity demands for services able to locally and privately validate user and provider requirements on new services to be invoked. At this purpose, in this dissertation, we propose a privacy-preserving framework for a choreographed service composition, where to enable privacy-preserving requirements evaluation [21].

1.2 Contribution

Inspired from motivations in Section 1.1, the dissertation aims at building a variety of secure protocols and application scenarios. Figure 1.1 depicts the major contributions of the dissertation. Every protocol meets three prerequisites on collaboration type, network model, and security. Let us describe one by one protocols and their application as follows.

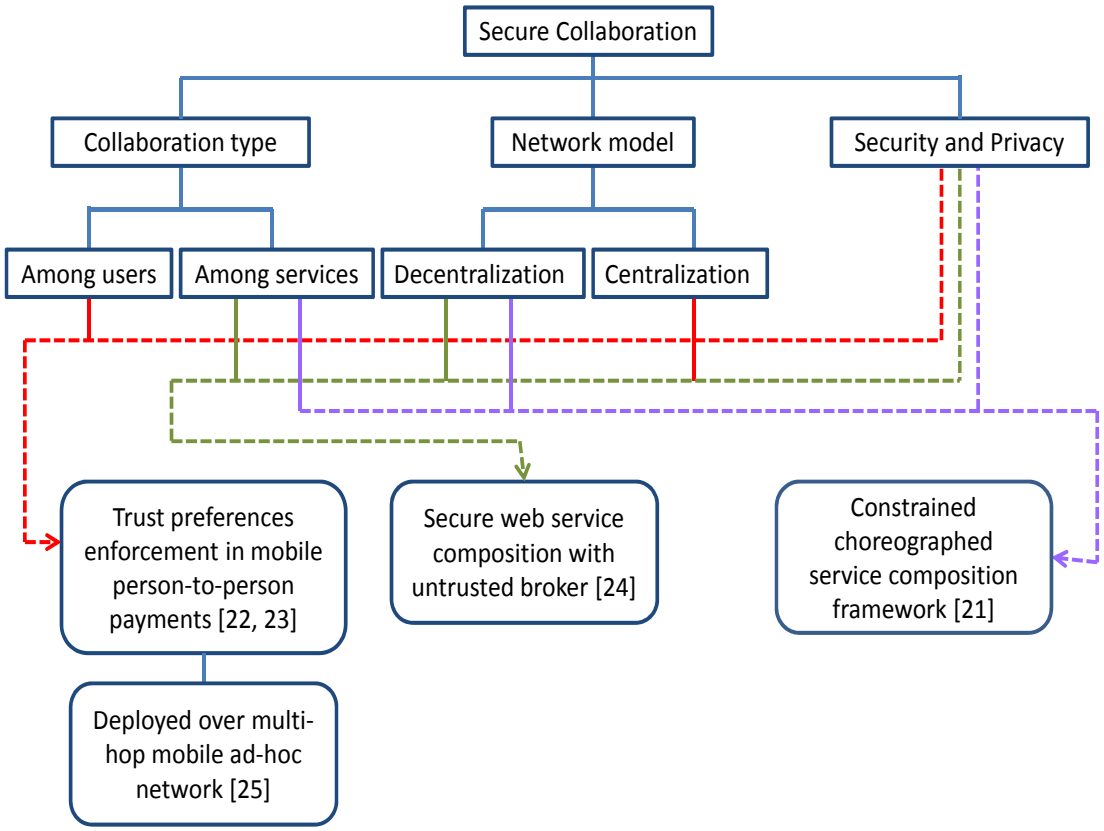


Fig. 1.1 Graphical representation of the main results of the dissertation

1.2.1 Scenario 1: Secure Mobile P2P Payment

We investigate a secure mobile social collaboration among multiple parties as motivated from Section 1.1 through the scenario of mobile P2P payment. In this scenario, we propose to make payer (i.e., payee) able to state a set of *trust preferences* stating how the potential payee/payer has to be connected with him/her in order to give/accept money. Trust preferences state conditions on: (1) the type of the relationship that must exist between the payer and the

payee (e.g., friend, parents, just-met), (2) the distance between the payer and the payee, i.e., the number of hops in the path connecting them, (3) the trust value associated with this relationship. As an example, a user can state that he/she will accept to transfer money only to payees that are friends of one of his/her friends with a high trust value (i.e., type=friends, distance=2, trust value = high). Before committing a money transfer, both payer and payee evaluate their local trust preferences to decide whether the other can be considered enough trusted.

Evaluating a trust preference implies finding connecting paths between payer and payee. In designing a protocol to support this task, we kept into account two main requirements. The first is that the solution has to support all existing mobile P2P payment methods as well as existing social network services, without forcing an integration between these platforms. Another relevant issue is that nowadays persons are used to join several social networks, maintaining different accounts with different contact lists. Therefore, we need to support path finding across different social network realms. To cope with these interoperability issues, we designed a solution such that trust preference evaluation is locally performed on each peer by means of a mobile application, called *SmartPay*, assuming that user contacts list is locally managed by the mobile application. [22, 23] Therefore, the main goal is to discover the relationship path between two arbitrary nodes in a distributed network. In designing our protocol, we have considered two essential requirements: (1) *relationship privacy* - relationship information is sensitive personal data. As such, information on the type, trust and depth of a given relationship should be known only by the users establishing that relationship, not by intermediate nodes participating in the protocol; (2) *limitations of mobile devices* - almost all mobile devices are characterized by low battery, low memory, low power, limited throughput. To cope with the above limitations, we propose of light cryptographic protocol for *mobile-oriented decentralized path discovery*. The protocol makes use of Elliptic Curve Cryptography (*ECC*) [72] to obtain a protocol running steadily and fast. To fully protect relationship information, we also propose a privacy-preserving approach to compute trust value between two adjacent nodes according to the well-known Tidal Trust algorithm [52]. Beside, we also propose an optimization strategy exploiting a anonymity technique, i.e., k-anonymity [136] so as to reduce the flooding of messages.

1.2.2 Scenario 2: Secure Mobile P2P Payment over MANET

In this scencario, we exploit social network relationships for P2P payments over MANET [25]. Inspired from benefits carried from MANET as mentioned in Section 1.1, this new communication means has open challenges that need to be addressed [126] [57]. Among them, the most relevant are: *(1) privacy and security*, *(2) energy efficiency*, and *(3) multicasting*.

The deployment over MANET of the mobile P2P payment protocol to deal with Scenario 1 has to cope with all these issues. Regarding privacy and security, the exchange of messages through MANET rises new challenges w.r.t. relationship privacy and limitation of mobile device as mentioned in Scenario 1. Indeed, according to the MANET protocol, a message might be forwarded through several mobile users before reaching the destination. Those users are not involved in the path traversal and thus do not have to infer any information about the payment transaction as well as aggregate relationship information. Hence, the main problem has to be cared about, that is, nodes in social graph can be malicious or honest-but-curious nodes trying to infer the aggregate relationship information. In this scenario, we present a solution suitable to MANET communication that provides a more secure protocol than the one in Scenario 1. Regarding issue (2), we use cryptographic algorithms on elliptic curves, that is, binary ECC [71] and Elliptic Curve Digital Signature Algorithm (ECDSA) [69]. These binary cryptographic algorithms make the mobile devices able to perform efficient computations and consume less energy. With respect to (3), we propose an optimization strategy exploiting secure comparison algorithms so as to reduce the flooding of messages in MANET [25]. In particular, our algorithm allows to evaluate encrypted relationship information inside a message to be forwarded in the network.

1.2.3 Scenario 3: Secure Orchestrated Web Service Composition against Untrusted Broker

This scenario investigates the web service composition in the centralized model based on the motivation presented in Section 1.1 [24]. As a reminder, in this model, the collaborators in this scenario comprise of a broker and several service providers. The broker coordinates service partners so as to carry out a sequence of activities requested by a user. In this case, the broker can read the data transferred from web service partners to it. To resist the broker and web partners on reading their unauthorized data, we propose a secure protocol based on the selective encryption of user credentials and service parameters. Our protocol ensures that the broker is not able to access these data as well as any information on invoked activities, whereas partner services are able to access only the portions of user credentials and service parameters generated by other partner services, needed for the correct execution of the assigned activities. Because user requirements are in terms of many conditions needed to be evaluated at web partners. Hence, the adopted encryption scheme allows the broker to evaluate test conditions on encrypted data, which is instrumental to enforce the correct execution order. In this dissertation, we also support more options for the expressive languages so that users can describe sufficiently their preferences.

1.2.4 Scenario 4: Privacy-preserving constrained choreographed service composition

This scenario investigates the decentralized collaboration among web services according to the decentralized model [21]. Each user can specify his/her requirements on an application. Then, the specified user requirements are changed to a workflow of activities. This workflow is done gradually through the peer-to-peer network. Because of the peer-to-peer nature of the choreography model, revising the way to enforce requirements is imposed. The user requirement enforcement basically needs two steps, including selecting the eligible service able to invoke the next activity in the workflow based on user requirements, then invoking the selected service. These two steps are required to be revised. There is a requirement regarding user privacy, that is, when user requirements move through the intermediate partners to their destinations, user requirements are validated privately at the intermediate services. User requirements can be read by only the authorized partners. Moreover, service providers have constraints on their partner candidates with whom they want to collaborate. For example, the requested service needs to use the same kind of encryption algorithm of the requesting service. For these purposes, we propose two secure protocols ensuring that service providers can locally and privately validate user and provider requirements. To the best of our knowledge none of them considers that the evaluation of these criteria requires to expose private information of users and providers, regardless of whom poses the requirements (i.e., users or providers) and the nature of these requirements (e.g., QoS, security, etc.). The only paper we are aware of dealing with these issues is [132], where a centralized and privacy aware approach for service selection of composite services has been proposed. Yet this scenario focuses on the decentralized model.

1.3 Dissertation Organization

This remainder of dissertation is organized as followed. We start with Chapter 2 for details of the recent works on the secure collaborative processes among multiple parties. We provide an overview of available collaborative protocols today, and point out the needs of complementing their deficiency. We focus more on discussing on the cryptography techniques applied in recent works, and spot the key problems to be considered in this dissertation. The details of the proposals for the dissertation is discussed in the rest of dissertation.

In Chapter 3, we present a secure decentralized collaborative protocol among users in mobile person-to-person payment [22, 23]. In this chapter we state privacy requirements,

then debate the possible cryptography algorithms applied into trust computing and evaluation. We also explain the security property of proposal against some attacks.

Chapter 4 investigates the communication environment, i.e., MANET, over which the mobile person-to-person payment protocol is deployed, and figures out the problems when running the protocol and solutions for them. Beside, we also describe security properties of the proposal, and highlight the optimization methods for the system performance [25].

Chapter 5 presents a secure centralized collaboration among services to provide a complex service to users [24]. In this chapter, we present the essential privacy points needed to be obtained, discuss the solutions and analyze the results achieved from the proposal.

In Chapter 6, we continue to investigate the collaboration among service, but now we study the decentralized model [21]. The different problems from the centralized model are expressed, as well as, the privacy and security points are detailed in this chapter.

The conclusion is made in Chapter 7. This chapter outlines the future plan as well.

Chapter 2

Related work

In this chapter we first introduce recent works on collaborative systems among multiple parties, so as to highlight the security and privacy risks in these systems. We then survey mobile P2P payment systems and the state of art in the web service compositions, particularly in regards of the security and privacy issues.

2.1 Multiple party collaboration

The first mass collaboration occurred as Linux source code was opened in 1991. At that moment, all software developers worldwide freely modified the Linux core and shared their contributions to the world. This improved much the quality of Linux services, and then Linux has had a long way to evolve until now. After that, in 2000, GoldCorp followed the same solution of Linux development team, they launched the problem Golden Challenge online to receive recommendations of social users on solutions for their gold mining crisis at that time.[1] They overcame the difficulty and have grown up from $100 million, of the time before Golden Challenge, to $9 billion. Following, the product Beehive of IBM, an internal social network, was released in 2007 for the cooperation among IBM employees worldwide. They carried out Beehive to promote their projects and gained the ideas from people in the company having different backgrounds. The collaborations worked out excellently with Beehive and brought back a huge benefit for IBM. In the meanwhile, the advent of web based collaboration occurred officially in 2001 when InnoCentive web based network was released. They used this product to make collaboration among their employees through web

[1] https://www.keithrozario.com/2012/07/opensource-gold-the-greatest-crowdsourcing-story-ever-told.html

browser.[2] InnoCentive helped them out of problems needing collaboration and gained the urgent solutions in time.

According to a recent survey about requirements on manipulating social collaborative techniques and tools, [8] the today social collaboration pervades and plays a key role in enterprises. We admit that the social collaboration puts an important contribution into improving solutions and developing the insight of organizations rapidly. That is a quick hop to reach a long distance ahead for enterprises, however, a company is meaningfully powerful not only because of the collaboration among their staffs but also for the customer services. As mentioned above, users, particularly social users, have requested the higher complexity on services. This pushes the online services to be upgraded and cooperated for survival. Microsoft also determine that the social collaboration carries much benefits for companies [103]. Not only social services need collaboration but web, mobile, and cloud services do as well.

Some today enterprise tools support social collaboration, to eliminate barriers to team productivity, and are launched to cloud. Microsoft cloud based tools[3] like Yammer and Lync make it easy for employees to connect, share ideas, and build teams, Office 365 breaks down barriers between departments, enabling collaboration from virtually anywhere, and Microsoft Dynamics CRM helps companies give customers a better, more responsive experience. With Yammer and Office 365, the global textile and apparel manufacturer Esquel Group increased productivity among its 7,000-member workforce resulting in an estimated $2 million in savings a year and facilitated new ways of knowledge sharing and innovation.[4] Beside Microsoft cloud product, blueKiwi[5] is one of leading providers of enterprise social collaboration software for companies of all sizes of diverse fields. It helps them connect with their customers through an enterprise social network and is hosted SaaS (software as a service) with the option of a Private Cloud configuration. One of big customer of blueKiwi is Atos. blueKiwi helps over their 60,000 employees in 4,000 active communities across 47 countries collaborate with each other daily to manage projects and workflow successfully. Beside, Cisco also have Web Collaboration products providing organizations with a tool to carry chances of collaboration among internal employees and with customers to facilitate the revenue.[6]

The requirements of social collaboration from enterprises now get higher and become a motivation to boost the researches to provide the good solutions to the world. However,

[2]http://www.forbes.com/sites/karlmoore/2011/09/15/from-social-networks-to-collaboration-networks-the-next-evolution-of-social-media-for-business/

[3]http://www.microsoft.com/enterprise/microsoftcloud/social/default.aspx#fbid=eUKomqXpXI8

[4]http://www.microsoft.com/enterprise/solutions/social_collaboration/default.aspx#fbid= eUKomqXpXI8

[5]http://www.bluekiwi-software.com/en/

[6]http://www.cisco.com/c/en/us/products/customer-collaboration/web-collaboration-option/index.html

under the extraordinary pushing power of using social collaboration tools, there are always another problems needed to be considered, such as, data security, user privacy, performance, etc.

2.2 Security and privacy in multiple party collaboration

In general, partners contribute their own information into the collaborative process and do not want the other sides to be aware of their personal information. As a report by Gartner, in 2014 the worldwide spent on information security $71 billion, an increase of 7.9% over 2013, with the data loss prevention segment recording the fastest growth at 18.9%.[7] IBM redefines requirements on network, data and user security and privacy along with pushing the quality of services [8] Cisco address risks regarding the security and privacy during collaboration, in terms of shared information, user identity, authentication, and authorization, etc. [29].

In the dissertation, we consider two kinds of attacker, that is, honest-but-curious (semi-honest) and malicious attackers. A party is called honest-but-curious adversary when it follows correctly the collaborative protocol agreed on by all participants in the protocol, but it might be curious about the private information in the computed result which it receives. In contrast, a malicious adversary party is more harmful, since he is not only trying to extract as much as private information of the participants, but also trying to change/intervene the result of computed data from collaborative protocol [55].

Risks by above adversaries are addressed in more detailed in some recent works. As in [10] [28], authors identify main risks coming up when some of parties have incorrect manner during the collaborative protocol enforcement. They might be honest-but-curious or malicious adversaries who try to extract the additional information from the shared content. Even though parties in the collaboration agree on the protocol about devoting their private information to others for the common computation, the main problem is that they do not have enough confidence in each others. Thus, they care about their shared data possibly disclosed to the others. Another issue is that the computed data outputted from the collaborative process can be then transferred to some parties. Some of those parties can infer the private information of other parties, or intervene the computed data of protocol incorrectly.

In order to preserve the privacy of parties in the collaborative protocol, some recent works [119, 114] exploit the anonymity techniques (i.e., including k-anonymity [136] and l-diversity [95]) to obscure users' personal information before collaboration. However, with

[7] http://www.computerworld.com/article/2598536/security0/security-spending-gets-boost-from-mobile–social-and-cloud–says-gartner.html

[8] http://www.isaca.org/groups/professional-english/cloud-computing/groupdocuments/ redefiningnetworks-forcloud.pdf

anonymity techniques, there is still the risk of disclosure of individual information. As in [113], authors spot the privacy disclosure when applying k-anonymity into solutions, after their experiments are done. One solution hides client's identity (e.g., its own IP address) by transfer a client's request through one or more intermediate proxies as done in onion routing systems such as Tor [36]. Hence, in some other recent work, the cryptographic mechanisms are considered to empower the user privacy and the data security. As in [35], authors use the homomorphic encryption, that is, RSA algorithm to allow n participants to cast their votes in a way that preserves the privacy of individual values, to be resistant to collusion even against as many as $(n-1)$ malicious insiders.

Some other works focus on multiple party collaborative computation. It relates to the computation done by all parties in the protocol but do not leak the private information of every side, for an example, two millionaires [158]. In particular, two millionaires want to know who is richer, without any of them revealing to the other his net worth. Authors in [96] deploy a generic computation for two parties to compute the common data, with no assumption of the trusted third party, using RSA, AES, and Elgamal. Some other works share the same idea can be counted as [27, 15].

Some recent works concern the privacy preserving of participants in calculating the intersection of private data, and do not want the other to know this private data. Freedman et al. [45] proposed a specially-designed secure multi-party computation protocol to compute set intersection between the input lists of two parties. It represents each party's inputs as the roots of an encrypted polynomial, and then it has the other party evaluate this encrypted polynomial on each of its own inputs. While asymptotically optimized for this setting, a careful protocol implementation found two sets of 100 items each took 213 seconds to execute (on a 3 GHz Intel machine) [46]. Kissner and Song [81] extended and further improved this polynomial-based protocol for a multi-party decentralized setting. Other works, as [45, 92, 82], authors also solve the same problem using secure set intersection.

Although, recent works put much effort in exploiting the cryptographic and anonymity techniques for their collaborative scenarios, their solutions cannot cover all cases. We believe that there are still many open scenarios needed to solve the same problem. In the dissertation, we describe respectively some of the open scenarios, by showing the proposed solutions designed in this thesis.

2.3 Decentralized social trust computing

The decentralized systems do not have the central controller for protecting data authentication and confidentiality, this causes an inherent insecurity and untrustworthiness [153]. Hence,

peers do the work of central controller. When several parties collaborate together for a common goal, they need to select the trustworthy partner satisfying all of their requirements so as to cooperate with. Hence, they need to quantify how much trust they have for the considered collaborator candidate. Actually, in the real society, people can appreciate others based on the information they experience from daily interactions in person. However, in the social network, people do not have face-to-face activities, everything are done in the virtual online community. This makes it tough to know which person is really trustworthy. As in a paper exploring a new perspective, namely the human capital approach, in understanding social trust and its formation, Glaeser et al. [51] said that *"Social trust in a group is equal to the perceived trustworthiness of a typical member or the average trustworthiness of all members, characterized by the proportion of cooperative players in the group. The level of social trust is determined by the distribution of cooperative tendency in the group."* It implies that the only thing based on which social users can rely on to assess people is the information recommended by a known user or the common social users, or shared from social user profile. The quantification of how much trustworthiness a person has for another user is so-called trust computing.

In order to visualize the interactive view of social network, a social network is mapped onto a social graph. This also eases the trust computing. A social graph might contain relationship type, trust level between every two users, and another information according to every specific goal.

More specifically, in the dissertation we model a social network as a directed labeled graph $\mathscr{G} = (V, E, RT, \phi)$, where RT is a finite set of relationship types, V is a finite set of users, $E \subseteq V \times V \times RT$ is a set of edges representing relationships between users, and $\phi : E \to [0,1]$ is a function mapping each edge $e \in E$ to a trust level. Given a relationship between two nodes v, v', denoted by $rel(v, v')$, this is defined as a direct relationship if $rel(v, v') = e \in E$, it is defined as an indirect relationship if $rel(v, v')$ connects the two nodes by a path consisting of more than one edge, all with the same relationship type. In case of indirect connections, the depth of a relationship is given by the number of edges in the connecting path.

Some works just focus on the trust computing metrics between two directly connected users. As an example, Velloso et al. [146] proposed a trust computing metric, for a node in mobile ad-hoc network to quantify its trust in its neighbor, by accumulating the previous personal experience and recommendations from its neighbors. The previous personal experience means that they keep track of behaviors of the neighbors to recognize that behaviors are good or bad, then assess the neighbor. Recommendations are the previous personal experiences transmission from neighbors. Authors propose the concept of relationship maturity based on the age of the relationship between two nodes. If the age of relationship is older, the

recommendation is higher, and vice versa. In [34], authors have the same approach to solve the same problem, the difference is that [34] has an description of operations on messages in mobile ad-hoc network, while [146] supports more parameters so as to make the trust quantification more exact. Moreover, some other works also share ideas based on concepts of previous personal experience and recommendation can be found in [56] [142] [89], [90]. In another work [26], authors combine the ideas of using recommendations and tracking the node behavior to compute the trust between two direct nodes and to evolve the trust in community. Authors then apply the metric to social network for two indirect nodes, by aggregating the trust of pairs of two nodes on the path connecting those two nodes. Golbeck et al. [54] propose a method for creating a trust network on the semantic Web by extending the FOAF schema. The model allows users to set a level of trust for people they know. The trust levels are sorted out from "distrusts absolutely" to "trusts absolutely". In the model, an ontology is used for defining different trust levels for the different contexts. Authors also concern the trust values between nodes not directly connected to each other by using trust-annotated FOAF[9] (the friend of a friend) graph. The trust is calculated with the weights on edges of the graph. In another work [53], Golbeck proposes TidalTrust, a metric of trust quantification using FOAF vocabulary and exploiting in social network. In this model, neighbors with higher trust ratings can agree with each other on the trustworthiness of a third party. TidalTrust supports two metrics of trust computing between two directly nodes and two nodes connecting through several intermediate nodes. There are also other works focusing on serving the decentralized networks, such as, [2, 56, 58, 70, 89]. Some other works have different approaches. [88] uses Bayesian to weigh the trust based on the occurrence time and exponential decrease method to set time to live on the old observation. Another work [84] computes the trust based on the probabilistic logic sampling.

The above works solve the need of nodes in validating the trustworthiness of a potential node before a collaboration. However, they do not consider preserving the privacy in computing the trust of a neighbor. As in [60], preserving a privacy in multiparty collaborative trust computing as: *"Let a be an agent that contributes its local feedback about an agent t, as part of a protocol to compute the reputation of agent t. Then the privacy of agent a is said to be preserved if during or after the execution of the protocol, no other agent in the system is able to learn agent a's local feedback."*. Some proposals for privacy preserving trust computation systems with an assumption that every node has a trusted hardware module. [77] requires every node a trusted platform module (TPM). TPM makes its owner node to prove that it is valid and a party in the system without leaking the node identity. Beside, it also

[9]FOAF vocabulary (http://xmlns.com/foaf/spec/) is one of the most common in semantic web. The vocabulary describes social users and their social network connections. It stores over 8,000,000 people, and used by many large web-based social networks.

allows the owner node to feedback privately and anonymously. Whereas, [157] and [147] both present a solution based on smart cards which are supposed to be trusted. Another work [78] tries not to use hardware modules, however, it needs an anonymous routing network. There are some other works operating with the aid of a centralized system, that is, [68] and [67], [6], all have the problem with the single point failure and the untrusted third party.

Therefore, in this dissertation, we discuss the need of a trust computation suitable for decentralized networks with the limitations of mobile environment and that to have independence of hardware or hardware plugins.

2.4 Secure mobile person-to-person payment

Section 2.4 is initiated with the introduction on recent well-known person-to-person payment systems in Section 2.4.1. In Section 2.4.2, we present a couple of famous systems in secure decentralized mobile payments. Then we discuss previous works on secure path discovery in Section 2.4.3. Finally, Section 2.4.4 ends Section 2.4 with a discussion about recent secure P2P payment systems over MANET.

2.4.1 Recent person-to-person payment systems

The first P2P payment emerged with Western Union service, and the next was Paypal [18]. These services really helped people in two distant locations making them able to transfer money to each other in safe and convenient. The two services follow the centralized model. They have been still well known and worked out until now. Based on the same target at P2P payment, but with different motivations, that is, to secure and comfort people in reducing to bring cash or wallet outside, and to balance the budget better, Kenya came up with the idea of mobile payment and banking.[10] Although Kenya was not the country inventing the mobile device, it was the first place giving birth to mobile payment. This idea turned into reality with M-Pesa, a mobile payment application, launched in 2007. In mobile payment, a conceptual monetary unit is used. An amount of such units represented for cash or check is stored in an application in mobile device, and certainly bought by device owners at corresponding stores. When making a payment, users show their mobile device near the card machine in a predefined distance. It is because most of mobile payment applications use wireless technologies like Bluetooth, Near Field Communication (NFC), etc., so they limit the distance. Since then, mobile payment applications have been spread all over the world. So far a half of all mobile money transactions in the world take place in Kenya, where

[10] http://www.techrepublic.com/article/the-worlds-unlikely-leader-in-mobile-payments-kenya/

annual transfers have reached $10 billion. One of well known mobile payment application revolutionizing the world of credit card cannot be missed, that is, Google Wallet,[11] released in 2011. Beside Google Wallet are VISA Wallet (multiple financial networks), ISIS (AT&T and T-mobile telecommunication providers), Serve (American Express Customers). Mobile payment promises to fetch an amount of money transfer to $670 billion by 2015.[12]

Though mobile payment promises to carry on tremendously growing up and overtakes the traditional payment by cash and check, the above mobile payment methods are practically needed in point-to-point transactions, e.g., supermarket/store payments. It means two points involved in a transaction obliged to locate close to each other in a predefined range, and still needs to go through the third party server. However, there is a need of dealing with a (in)direct payment between two arbitrary points in a undetermined distance without going through the third party server. This leads to the advent of mobile P2P payment. It is possible to find very interesting initiatives, like Zopa,[13] Ratesetter[14] and Funding Circle.[15] Another notable example of P2P payment system is given by Ripple [121], which is a protocol for an open source payment system.

2.4.2 Security-aware decentralized mobile payment.

Let us first introduce a recent typical leading system in decentralized e-cash systems, namely Bitcoin. Bitcoin [108] was published in 2008 by Satoshi Nakamoto, and nowadays becomes a famous innovative peer to peer payment with no central authority or bank. This virtual currency system has been the first one accepted by the widespread popularity. Let us go details of this payment method.

In Bitcoin, an electronic Bitcoin coin is defined as a chain of digital signatures. Each Bitcoin user is referenced in each transaction by a pseudonym as a Bitcoin address. Each Bitcoin address is attached to a unique pair of public key and private key. These keys are used for transferring the ownership of the Bitcoin coin among addresses. When an owner receives a Bitcoin coin, then makes a digital signature of a combination including the previous hash and the public key of the next owner, and appends this signature to the coin before sending its updated version to the next owner. The final owner receiving the coin, called payee, can validate the ownership by verifying the chain of signatures. A problem emerges in case a Bitcoin coin spends double. To overtake this problem, a hash based proof-of-work scheme is implemented to generate blocks and widely publish them. A block includes a hash of the

[11] http://www.thinslices.com/mobile-payment-apps/
[12] http://www.bonsoni.com/news/research-shows-mobile-phone-payment-double-by-2013/
[13] http://zopa.com/, http://uk.zopa.com/about-zopa/peer-to-peer-lending
[14] http://www.ratesetter.com/
[15] https://www.fundingcircle.com/

previous block, a value nonce, and other requested data users want to include. A Bitcoin user needs to search in the block for a nonce, a value beginning with a number of zero bits, when hashed, the result is below a given value. If such a nonce is found out in the block, it is included in a new block. The updated block is forwarded to all users in the network to check its correctness by verifying the hash. Each block links to the previously generated block, the Bitcoin block chain grows in the network. Bitcoin relies on this mechanism to resist double-spending attacks.

However, in another work by Gervais et al. [50] addressed that today Bitcoin is not a fully decentralized system as promised. Authors said that the original version of Bitcoin aimed truly at a fully decentralized system, yet recent versions cannot maintain such a feature. They are hosted and managed by central services, and a considerable share in Bitcoin market is noticeable. Moreover, Bitcoin depends on its developers that can retain privileged rights in conflict resolution and maintenance of the official client version.

Meanwhile, another recent work, namely Zerocoin [104] addresses risks of using pseudonym of Bitcoin as an identifying address. In particular, Zerocoin deduces that Bitcoin is secure relied on two assumptions that its nodes are honest and its proof-of-work can deter Sybil attack [40], but all transactions are public and the anonymity of user's addresses is protected by the pseudonym technique. Hence, an attacker can identify participants of Bitcoin coins partially or completely. Zerocoin proposes a solution to break the link among participants without using any trusted parties. Let us describe the intuition behind Zerocoin scheme. Zerocoin supposes to use a physical bulletin board. To mint a coin, each owner needs to generate a serial number S and digitally signs S for a commitment. This can reveal the owner identity. The commitment results in a coin C and a number r. To hide S when using C, the owner can only use r to retrieve C. This step can break the link between the owner's identity and the currency. Then, the owner can public C to other users by pinning C on the bulletin board, the other users accept C if it is structured correctly then calculate the sum of currency. To redeem C from bulletin board, the owner of C needs to generate a value π relating to r, and posts it to the bulletin board. The other users check if the coin has been spent in the previous transactions so as to decide if the requesting owner can spend the coin or not. Not to use the bulletin board and to launch Zerocoin to the fully decentralized model, the idea is to apply the block chain in Bitcoin as a Zerocoin's bulletin board. In Zerocoin, authors use RSA cryptography, with 3072 bit as key size at least. This can prevent attacker in attempting to infer the identity of coin owner from the digital signatures, but Zerocoin cannot be effective if it goes mobile when RSA consumes much main memory in computing as well as space to store the generated cipher texts. This makes Zerocoin unable to surpass the limitations of mobile environment.

2.4.3 Secure path discovery

In [38], authors propose a decentralized solution for path discovery based on public key cryptography, but the user triggering the path discovery can learn all properties of each relationship between two nodes in the path. To overcome this drawback in [39] authors enhance the protocol forcing users to be able only to access aggregated value of relationship properties. [39] assumes that each node sets a trust level on its contacts, and adopts the probabilistic multiplicative privacy homomorphism for computing the rational trust and depth of two direct or indirect nodes. This supports a node, owning a resource and setting an access control[16] on it, in evaluating another node requesting its resource. For example, there are three nodes in the protocol, i.e., A, B, C with B locating between A and C. The trusts, respectively between A and B, B and C, are T_{AB} and T_{BC}. The total trust, calculated at B and sent to C, is $T_{AC} = T_{AB} \otimes T_{BC}$, after applied with homomorphic, it results in $E_{P_C}(T_{AC}) = E_{P_C}(T_{AB}) \otimes E_{P_C}(T_{BC})$ where P_C is the public key of C, known to A and B, and $E_{P_C}()$ is the probabilistic homomorphic encryption function. As a consequence, this work is successful in aggregating trust and depth on the connecting path between two nodes. The missing things are that (1) this work publishes the public key of the destination node, which does not preserve the destination user's privacy, (2) they do not validate the relationship type of the entire connecting path, (3) the data set used for computing trust and depth is not the social data, but locally set up and stored in every node by itself, so the data on the entire connecting path seems more subjective. Moreover, given the requested mobility factor, this work does not go mobile, so authors do not concern properties of homomorphic encryption algorithms affecting to the space and pace load if it is working in mobile network.

In [101], authors focus on calculating the depth between two indirect nodes, i.e., length of the path connecting the two nodes. Like [39], [101] does not support calculating trust and relationship type of the connecting path, as well as does not exploit the social data. However, this proposal is fully decentralized with no third party. In particular, while discovering the path connecting those two arbitrary nodes, the protocol uses a hash function to aggregate the depth without revealing the relationship information of nodes on the path. The path discovery targets the way a node evaluates another node in the network. In the work, assume that there is a path in the network as $A \rightarrow B \rightarrow C \rightarrow D$, node A wants to verify the trust between A and D. A generates a random value, called r, known to only A. A hashes r by a cryptographic algorithm and obtains $T_1 = H(r||1)$ where 1 is the order of the branch from A to B. A then sends T_1 to B, B calculates the new hash, called $T_2 = H(T_1||1)$, then forwards it to C. C hashes the received token again and gains $T_3 = H(T_2||1)$. If C has more than one branch, the

[16] Access control is a set of conditions specifying compulsory requirements all which users need to satisfy to use services/resources.

number following the previous hashed value in the hash function is increased for every node on every branch. This work first catch its objective in calculating privately the depth between two (in)direct nodes against some honest-but-curious and malicious attacks wanting to know the information in the token. The missing things in the work are (1) authors do not remind a specific hash function, (2) the work do not mention how the destination node, i.e., the one supposed to be the destination of the token, can evaluate the received token, (3) the work limits in computing the depth and does not mention anything relating to the relationship type and the trust. In addition, it is ineffective for a real large social network because of the initial required token flooding phase.

Shared the idea with [39] and [101], another work [156] proposed a privacy-preserving fully-decentralized protocol, namely P^3D. The protocol uses a data structure, called token, sent from a source node to a destination node. While moving through the network, the token aggregates the relationship information on its path. [156] overtakes [39] and [101] due to followings: (1) authors exploit the social data for calculating the trust between two arbitrary nodes, this makes data more real and objective; (2) the protocol can compute aggregate trust, depth and relationship type of the connecting path. Authors compute privately depth and trust using logarithm based hash function, and compute relationship type using Elgamal encryption. Moreover, if [101] limits the computation in a specific length, which makes it unable to be applied to a real social network, as well as faces a problem of token flooding, [156] can deal with those issues by proposing a computation method bounding the result, and a strategy of optimizing token flooding. [156] can resist on the honest-but-curious attack. As a consequence, [156] achieves the privacy during the progress of discovering the path.

With our observance, all above works operate very well on the wire network, e.g., Internet, since physical resources are sufficient for computation. However, to launch them to mobile network, we need to take care of another limitations of mobile environment, such as, limitations of mobile devices in power, memory, energy, and of mobile network bandwidth.

2.4.4 Security-aware MANET payment

Authors in [61] propose a secure electronic payment system over MANET, called PayFlux. PayFlux applies SPKI (Simple Public Key Infrastructure) to improve the performance of processing encryption/decryption. However, PayFlux only works with direct connections (i.e., one hop) between the payer and the payee. In [66], [91], authors proposed a secure payment protocol for a vehicular ad-hoc network between a client and a merchant. [66] uses symmetric cryptography whereas [91] exploits ECC and hash functions. These aim at improving the performance, while still preserving user privacy and data security. However, [61] [66] [91] focus on direct connections and e-cash transfers, whereas, our work considers

both direct and indirect connections and makes trust preference enforcement available for users to decide a money transfer.

Authors in [120] propose the P2P payment in multiple hop wireless networks as well. Each node in the protocol is assigned a trust value. A packet of payment request is routed through the network according to these trust values. It implies that the packet moves through nodes having the high trust values. After every successful transaction, these trust values are updated. Whereas, our protocol works on relationship information of every user retrieved from their social network sites. Beside, [120] deploys the decentralized-based framework in wireless network, however, they still need the third party to authenticate the data. Meanwhile, we do not need the support of any third party, our proposal is a fully decentralized model. Moreover, they use only hash function for data authentication while our proposal makes data more confidential and authenticated as well by using both hash functions and encryption algorithms. Hence, their data security is not guaranteed as the one in our proposal. In our proposal, intermediate nodes are not let to know the source and the destination node to avoid the trace made from a malicious intermediate node while the intermediates in [120] can know the source and the destination nodes.

In [102], a local secure scheme supports nodes in evaluating the trust between two nodes, node i and node j, in an ad hoc network, when node i requests node j for a communication. In particular, node i retrieves a set of common connected nodes of node i and node j, then requests common nodes to send to node i their reputation recommendation on node j. To retrieve this intersection and to let each node in the protocol unaware of the set of nodes of the other side, author adopted the homomorphic Paillier's encryption and polynomial intersection calculation.

2.5 Web service collaboration

A web service is a software system designed to support interoperable application-to-application interactions over the Internet. One of the major goals of web services is to make easier their composition to form more complex services. This leads to a requirement on collaboration among services, called a web composition. The two main techniques for web service composition are respectively orchestration and choreography. According to [99], orchestration and choreography models are defined as followed.

"In orchestration, the involved web services are under control of a single endpoint central process (another web service). This process coordinates the execution of different operations on the Web services participating in the process. The invoked Web services neither know and nor need to know that they are involved in a composition process and that they are

playing a role in a business process definition. Only the central process (coordinator of the orchestration) is conscious of this aim, thus, the orchestration is centralized through explicit definitions of operations and the invocation order of Web services".

Each orchestrated web service is specified under an automated workflow of activities using a particular executive language, such as Business Programming Executive Language (BPEL) [4], and a BPEL engine [110], i.e., the coordinator of the orchestration to execute the workflow at runtime. An orchestrated workflow is typically exposed as a service that can be invoked through an API. It does not describe a coordinated set of interactions between two or more parties.[17] Recent works follow orchestration model as [33, 106, 65, 75].

In contrast, *"choreography does not depend on a central orchestrator. Each Web service that participates in the choreography has to know exactly when to become active and with whom to interoperate. Choreography is based on collaboration and is mainly used to exchange messages in public business processes. All Web services which take part in the choreography must be conscious of the business process, operations to execute, messages to exchange as well as the timing of message exchanges" [99].*

Choreography refers to a description of coordinated interactions between two or more parties. For example, you request a bid, I return a quote, you submit a purchase order, I send you the goods.[18] Recent works follow choreography model as [5, 149, 73].

Each of technique has own advantages and disadvantages. It depends on the specific requirements of the application, the more suitable technique should be chosen [124] [130]. As discussed in [131], these requirements can be posed both by users requiring the composed service as well as by the companies provisioning the atomic services. In general, requirements can refer to several dimensions, such as Quality of Service (QoS) [86] composing of business value quality, service level measurement quality, interoperability quality, business processing quality, manageability quality.[19] or security quality [24], as services might have strong security requirements on services with which they have to cooperate during the service composition deployment.

2.5.1 Security-aware orchestrated web composition

Orchestration has a BPEL engine as a central mediator to control a static sequence of activities described in a BPEL-based file. Such an approach is simpler and easier to be implemented, making orchestration a valid solutions for composite web service deployment. Literature presents several approaches for orchestrating constrained web service compositions, where

[17] http://www.infoq.com/news/2008/09/Orchestration
[18] http://www.infoq.com/news/2008/09/Orchestration
[19] http://docs.oasis-open.org/wsqm/WS-Quality-Factors/v1.0/WS-Quality-Factors-v1.0.html

QoS [94, 148, 128] and/or security criteria [24] [131] have been considered. However, to the best of our knowledge none of them considers that the evaluation of these criteria requires to expose private information of users and providers, regardless of whom poses the requirements (i.e., users or providers) and the nature of these requirements (e.g., QoS, security, etc.). One paper we are aware of dealing with these issues is [132], where a centralized and privacy aware approach for service selection of composite services has been proposed. As such, we assume the workflow underlying the business process is encoded into a BPEL document and processed by a server hosting a BPEL engine. According to the orchestration paradigm, the broker coordinates the invocation of services involved in the composition, i.e., partner services, by passing the needed parameters. In general, all previous proposals for the service orchestration model consider the broker as a trusted entity. As such, they never paid attention to the fact that the broker is able to access several pieces of sensitive data, such as: the data given as input by the user invoking the composite service (hereafter credentials), the final outcome of the composite service, as well as internal parameters (i.e., variables and values generated as output by partner services).

As we mentioned above, to the best of our knowledge, we are not aware of work on untrusted BPEL engines for managing an orchestrated web service composition. However, there are some proposals on untrusted servers but in different scenarios. For instance, Feldman et al. [43] proposed SPORC, a generic framework for building a variety of collaborative applications with untrusted servers. An encrypted document is stored on the untrusted server and authorized users can access it. Tople et al. in [144] investigated the feasibility of a web server architecture where the vulnerable server virtual machine (VM) runs on a trusted cloud provider. All sensitive web content is made available to the vulnerable server VM in encrypted form. However, both [43] and [144] are not targeted to orchestrated web service composition.

However, the BPEL engine needs to follow a workflow pattern [152] [1] to perform a web service composition. As in [13], authors presented a secure framework in the sequence workflows. Blundo et al. in [14] presented a framework to authenticate a secure workflow by addressing a recent cryptographic tool and aggregate signatures to validate the orchestration by requiring all partners to sign the result of their computation. [14] can authenticate partner web services joining the workflow, but cannot preserve the privacy and the security of user and data. In another work [115], authors proposed Blindfold, a novel system that enables users to upload encrypted data to the server and to search it without revealing the true keys or values to the server or third parties. [115] supports user to search the encrypted values.

2.5.2 Security-aware choreographied web composition

To cope with weak points of the centralized model, recently many works have dealt on web service selection using the decentralized model. Choreography is a decentralized model with a dynamic sequence of web services used for B2B (Business-to-Business) applications, but it is hard to be specified and implemented. In this model, peers share the service reputation/trust levels together and manage them locally, so it can solve the problem of the single point as well as the resource requirements for a central node. Therefore, in choreographed web service composition, each peer has two key acts, that is, selecting a eligible service able to collaborate with it, and invoking the selected. This model is also considered as an effective solution to deal with problems of mobility, fault tolerance, reliability and the ultra large scale of the Future Internet [3]. To select a good service meeting all of its requirements, a peer often needs to consider many criteria on both functional and non-functional requirements, such as, response time, availability, throughput, security, reliability, and execution cost [12].

In [151] authors proposed a super-agent based framework where super agents share honestly their reputation information, and can also build the global reputation from public's opinions, as well as, form agent communities with similar interests. Also being in this tendency, we build up a choregraphy-driven web service selection solution but do not make available agent communities as a service. Our proposal is to select and execute at run time web services which match a list of requirements in a predefined work flow. In another related work [3], Ahmeh et al. proposed a selection choreography. This work calculates the time of the supported functions of a web service as a metric while our proposal considers in details at the parameters of a service candidate. In [64], authors presented how to use probability to determine which service can enforce the next activity in the work flow.

However, with all above decentralized based approaches, there are still risks for data exchanged among web services, as an example, parameters can be read by unauthorized services. Recent works having focused on the data security can be listed as [9] [111]. In these works, IBM and Oracle support fundamental security tools for users (i.e., between two parties in a transaction) to apply into their protocol designs relating to web services. They do not support a specific security solution for web service selection and composition. Particularly, authors in [9] used the SSL technique and authentication and username tokens to protect the data, and in [111] authors supported the web service security at three level (that is, message level, transport layer, and access control layer). Moreover, another approaches [117] [118] made use of a set of security patterns and rules to determine the security properties that should be preserved by individual services so that the security properties of the overall composition can be satisfied too. In [20] authors proposed a framework following which users can reach a secure web composition, they quantified the amount of trust and security based

on the defined QoS-based confidentiality properties. In another approach using cryptography scheme, authors in [98] applied the asymmetric cryptography to the exchanged messages into the web composition in the orchestration model. Hence, the cryptography application is simple just for the authorized parties to enable to read their messages.

As our observance, recent works have not worked on selecting an efficient and secure service in the decentralized model, or securing the data transferred between services. Hence, in the dissertation we would like to process a complete combination of selecting an efficient service and invoking it locally, privately and securely. Beside, we consider the progress in which peers exchange their private performance information to evaluate their potential partner for a collaboration, and in which peers evaluate the functional requests to see if they can respond those requests also in a secure and private way. As our best knowledge, this is a new open problem and will be presented in more details in the upcoming chapters.

Chapter 3

Mobile Person-To-Person Payment

Today we are experiencing the rise of new digital methods of making person-to-person (P2P) payments that, in contrast to traditional checks and cash, allow customers to use their mobile phone to support P2P payment. As an example, Gartner [47], forecasts that in 2016 there will be 448 million mobile payment users in a market worth 617 billion USD, and the global mobile transaction volume and value averages 42% annual growth between 2011 and 2016.

Despite the success of these new digital P2P payment methods, we believe that, to fully enable this increasing rise of digital wallets, relevant challenges still need to be addressed. As a step in this direction, this chapter presents our proposal that leverages social network connections to help the decision making processes behind mobile P2P payments. Indeed, in real life, we make decisions on accepting money from or on giving money to someone based on what we know about him/her. More precisely, we propose to make payer (i.e., payee) able to state a set of *trust preferences* stating how the potential payee/payer has to be connected with him/her in order to give/accept money. Trust preferences state conditions on: (1) the type of the relationship that must exist between the payer and the payee (e.g., friend, parents, just-met), (2) the distance between the payer and the payee, i.e., the number of hops in the path connecting them, (3) the trust value associated with this relationship. As an example, a user can state that he/she will accept to transfer money only to payees that are friends of one of his/her friends with a high trust value (i.e., type = friends, distance = 2, trust value = high). Before committing a money transfer, both payer and payee evaluate their local trust preferences to decide whether the other can be considered enough trusted.

Evaluating a trust preference implies finding connecting paths between payer and payee. In designing a protocol to support this task, we kept into account two main requirements. The first is that the solution has to support all existing mobile P2P payment methods as well as existing social network services, without forcing an integration between these platforms. Another relevant issue is that nowadays persons are used to join several social networks,

maintaining different accounts with different contact lists. Therefore, we need to support path finding across different social network realms. To cope with these interoperability issues, we designed a solution such that trust preference evaluation is locally performed on each peer by means of a mobile app, called *SmartPay*, assuming that user contacts list is locally managed by the mobile app. Therefore, the main goal is to discover the relationship path between two arbitrary nodes in a distributed network. In designing our protocol, we have considered three essential requirements: (1) *relationship privacy* - relationship information is sensitive personal data. As such, information on the type, trust and depth of a given relationship should be known only by the users establishing that relationship, not by intermediate nodes participating in the protocol; (2) *limitations of mobile devices* - almost all mobile devices are characterized by low battery, low memory, low power, limited throughput. To cope with the above limitations, we propose of light cryptographic protocol for *mobile-oriented decentralized path discovery*. The protocol makes use of Elliptic Curve Cryptography (*ECC*) [72] to obtain a protocol running steadily and fast. To fully protect relationship information, we also propose a privacy-preserving approach to compute trust value between two adjacent nodes according to the well-known Tidal Trust algorithm [52].

The remainder of the chapter is organized as follows. Background and notations are presented in Section 3.1. Section 3.2 introduces trust-driven mobile P2P payments. Mobile-oriented decentralized path discovery protocol is presented in Section 3.3. A method of optimizing the message flooding is described in Section 3.4. Security properties of the proposal is presented in Section 3.5. Experimental results show the effectiveness and efficiency of the proposal in Section 3.5. We describe the time complexity of proposal in Section 3.7.

3.1 Background and notations

3.1.1 Trust preferences in Social Networks

We model a social network as a directed labeled graph $\mathscr{G}$=(V,E,RT,ϕ), where RT is a finite set of relationship types, V is a finite set of users, $E \subseteq V \times V \times RT$ is a set of edges representing relationships between users, and $\phi : E \rightarrow [0,1]$ is a function mapping each edge $e \in E$ to a trust level. Given a relationship between two nodes v, v', denoted by $rel(v,v')$, this is defined as a direct relationship if $rel(v,v') = e \in E$, it is defined as an indirect relationship if $rel(v,v')$ connects the two nodes by a path consisting of more than one edge, all with the same relationship type. In case of indirect connections, the depth of a relationship is given by the number of edges in the connecting path. Literature offers several algorithms to combine

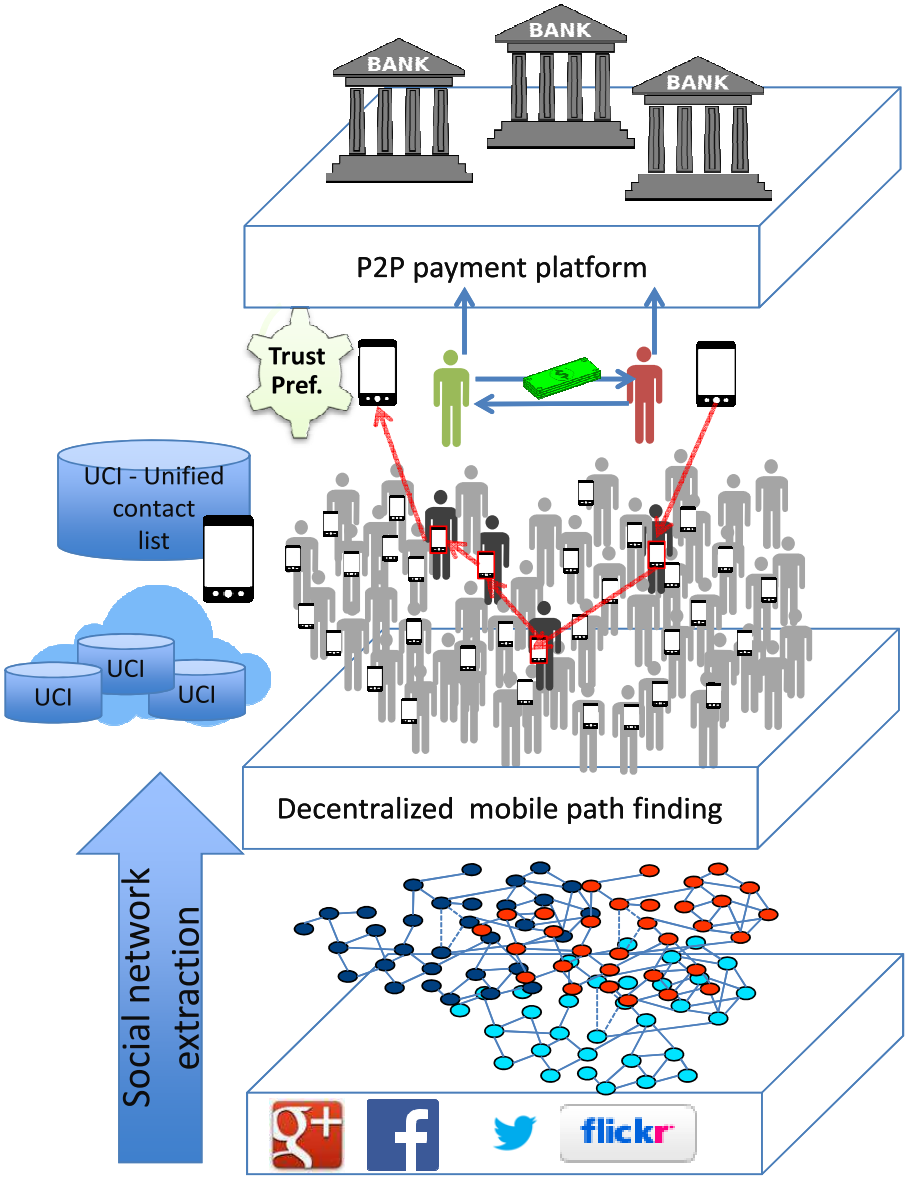


Fig. 3.1 Trust-driven mobile Person-to-Person payments

and to aggregate the trust values associated with the edges involved in a connecting path so as to compute trust value for an indirect relationship. In this protocol, we adopt one of the most well-accepted one, that is, Tidal Trust [52] (cfr. Section 3.3).

Based on this notation, we can now formalize *trust preferences*. A trust preference is defined as a pair (P, TC), where P indicates whether the trust preference has to be applied for a money transfer where the specified user is the payer (P=*payer*), or the payee (P=*payee*), and TC is a set of *trust conditions* specifying trust requirements. In particular, a trust condition is a tuple $TC = (rt, d_{max}, t_{min})$, which states that between the trust preference owner and the other party there should exist a relationship of type rt, with maximum depth equal to d_{max} and a trust level greater than or equal to t_{min}.

For simplicity, in the chapter, we assume that a trust preference consists only of one trust condition TC, and we denote with rt_{TC}, $depth_{TC}$, and $trust_{TC}$ the conditions specified on relationship type, depth and trust in TC.

3.1.2 Elliptic Curve Cryptography (ECC)

In our protocol, we use Elliptic Curve Cryptography (*ECC*) [72] based on the binary finite field F_2^m. As such, we assume that each node v_i using *SmartPay* has a pair of keys (k_i, $k_i \cdot B$), where the public key $k_i \cdot B$, the Elliptic Curve (E) and the base point B will be publicly published when v_i installs *SmartPay*, whereas the private key (k_i) is known only by v_i. Let us remind *ECC* encryption/decryption operations by assuming that Alice wishes to send a message M to Bob, and Bob has a pair of keys (k_{Bob}, $k_{Bob} \cdot B$), where k_{Bob} is Bob's private key, and $k_{Bob} \cdot B$ is Bob's public key. To encrypt M, Alice generates $E(M) = (Y1, Y2)$, $Y1 = r_{Alice} \cdot B$, $Y2 = M + r_{Alice} \cdot (k_{Bob} \cdot B)$, where r_{Alice} is a random integer generated by Alice. Given $E(M)$ Bob uses his private key k_{Bob} to retrieve M as follows: $D(E(M)) = Y2 - k_{Bob} \cdot Y1$.

3.2 Trust-driven mobile P2P payments

As depicted in Figure 3.1, *SmartPay* implements the bridge between social networking services and P2P payment services. We assume that *SmartPay* periodically synchronizes with user social network accounts so as to extract their contact lists. These are merged into a unique list, called Unified Contact List (UCL). We assume that users UCL can be either locally stored or stored into a cloud public storage service.[1]

[1] Protection of UCLs at cloud side is out of the scope of the work presented in this chapter. However, existing encryption techniques for cloud (e.g., [140], [105]) can be easily adopted to make only authorized mobile apps able to access the proper UCLs.

SmartPay provides users with the possibility of further classifying their relationships by specifying, for each of them, the type of the relationship, e.g., parent, colleague, classmate, defined according to a standard vocabulary like FOAF [19], and a trust value, representing the strength of the relationship. As described in Section 3.3.2, *SmartPay* also provides a tool for suggesting a user a proper trust value for those users he/she does not know, based on the well-known Tidal Trust algorithm. Thus, given the UCL, *SmartPay* can access the user social graph, representing the merge of the partial views the user has on each single social network. Formally, the social graph follows the definition and notation provided in Section 3.1.

Before committing to a money transfer, a user evaluates through *SmartPay* whether the payer/payee satisfies his/her local trust preferences and, if this is the case, the user proceeds with his/her preferred mobile P2P payment method. This evaluation requires to discover relationship paths connecting the two involved nodes, to verify if at least one of them satisfies the specified trust preferences. Since relationship information is locally managed, we propose a distributed protocol that, by means of node collaboration, traverses the social graph and looks for connecting paths. For each of the traversed path, the protocol collects the information on trust, depth, and relationship type. The collected information is stored into a set of data structures (called *tokensets*) and propagated through the graph. For simplicity, hereafter, we assume this process is always started by the payer. Therefore, once the payer receives a request of a money transfer from a payee through an email or a message on the social network, the payer initializes a tokenset including three distinct tokens, one for each path property (i.e., depth token, trust token, and relationship type token, respectively). The payer then forwards the tokens to his/her direct neighbors. These nodes update the received tokenset with the information on depth, trust and relationship type of the edge to be traversed and forward the updated tokenset to their neighbors. The process is iterated until the payee receives the tokenset, which is then send directly back to the payer by email or mobile internet. As soon as the payer receives back the set of tokensets, he/she retrieves the specified trust condition `TC`, i.e., the trust condition in the trust preference defined for the payer role. Then, by using the relationship type token, the payer is able to determine whether all the edges in the path have a relationship type equal to the one required by the trust condition. In case the edge relationship types are different, the payer is not able to determine their real values. In contrast, by exploiting depth and trust tokens the payer can determine the number of edges, and the trust value of the indirect relationship, without knowing the trust value of any edge in the path.

3.3 Mobile-oriented Decentralized Path Finding

In this section, the protocol will be described in further details introducing how tokens are initialized, updated and propagated along the graph. Before that, we present notations we adopt hereafter. According to the social network model given in Section 3.1, a path p traversed by the collaborative protocol can be denoted as $p = \{v_0, v_1, \ldots, v_p\}$, where v_0 is a payer and v_p is a payee. Thus, given p we denote with $Tr_{(i)(i+1)}$, $rt_{(i)(i+1)}$ and $d_{(i)(i+1)}$, respectively, the trust value, relationship type and depth of the edge connecting v_i to v_{i+1} in p. Moreover, the proposed protocol exploits *ECC* to protect relationship privacy. Data to be encrypted with *ECC* have to be encoded in points in elliptic curve.[2] Therefore, given an edge e=(v_i,v_{i+1}), we denote with $P_{d_{(i)(i+1)}}$, $P_{rt_{(i)(i+1)}}$, $P_{Tr_{(i)(i+1)}}$, respectively, the points in (E) encoding depth, relationship type and trust value of edge e.

3.3.1 Depth computation

By exploiting *ECC* properties, depth of the traversed path p can be computed by securely aggregating points in (E) encoding the depth of each single edge in p. Thus, each node v_i generates a big random number $r_{(i)(i+1)} \in \mathbb{N}^+$, which is used together with the payer's public key ($k_0 \cdot B$) to encrypt the distance between v_i and v_{i+1}, that is, the value 1. The obtained encrypted value is then added to the content of the received depth token. It is important to note that, even if depth values have always the same value (i.e., 1), analysis of encrypted text from attackers is still hard since each user in the path selects a different big random variable $r_{(i)(i+1)}$. Moreover, only the payer v_0 can know the final depth value of the whole relationship path, since intermediate nodes only participate in aggregating the depth value in the relationship path without the ability to decrypt it. Thus, the privacy of the payer and the previous nodes is preserved.

Let us see the depth token computation by presenting its initialization, computation and validation.

Depth token initialization. The payer v_0 generates the point $P_{d_{(0)(1)}}$ on (E) corresponding to $d_{(0)(1)}$. Then, it randomly generates a big number $r_{(0)(1)}$ for encrypting $P_{d_{(0)(1)}}$ with his/her public key $k_0 \cdot B$. We obtain: $E_{(0)(1)} = (Y1_{v_0},\ Y2_{v_0}) = (r_{(0)(1)} \cdot B,\ P_{d_{(0)(1)}} + r_{(0)(1)} \cdot k_0 \cdot B)$. The resulting encryption is then broadcasted to each of his/her direct contacts, that is, the node v_1 in each path p the protocol will find.

[2] Literature offers several algorithms for point compression (see [79] [80] [42]) that can be used to encode and decode a bit string in F_2^m into a point on a binary elliptic curve (E). These mapping functions do not affect the security of the protocol at all. For this reason, in this work a message W is mapped to a point on (E) as $P = W \cdot B$, where B is the base point, due to the simplicity and the additive property of the mapping.

Depth token computation at intermediate nodes. Once an intermediate node v_i receives the depth token from a neighbor v_{i-1}, it has to update and send it to each of its neighbors, except from v_{i-1}. Each forwarded depth token has to be updated so as to add to its contents, i.e., $E_{(0)(i)}$, the value $E_{(i)(i+1)}$, that is, the encryption with v_0 public key of point $P_{d_{(i)(i+1)}}$, where $d_{(i)(i+1)}$ is the depth between v_i and neighbor v_{i+1} to which the token will then be forwarded.

As such, we have that: $E_{(0)(i+1)} = E_{(0)(i)} + E_{(i)(i+1)}$, where $E_{(i)(i+1)} = (r_{(i)(i+1)} \cdot B, P_{d_{(i)(i+1)}} + r_{(i)(i+1)} \cdot k_0 \cdot B)$. Thus, $E_{(0)(i+1)} == (r_{(0)(i)} \cdot B + r_{(i)(i+1)} \cdot B, (P_{d_{(0)(i)}} + r_{(0)(i)} \cdot k_0 \cdot B) + (P_{d_{(i)(i+1)}} + r_{(i)(i+1)} \cdot k_0 \cdot B)) = ((r_{(0)(i)} + r_{(i)(i+1)}) \cdot B, (P_{d_{(0)(i)}} + P_{d_{(i)(i+1)}}) + (r_{(0)(i)} + r_{(i)(i+1)}) \cdot k_0 \cdot B)$.

Then, v_i broadcasts the resulting $E_{(0)(i+1)}$ to each of its direct neighbors, i.e., v_{i+1}:

Depth token validation. The depth token is flooded until it reaches the payee v_p. Then, for each founded path, v_p sends the corresponding depth token back to the payer v_0. By the depth token, payer v_0 obtains the aggregated depth value $E_{(0)(p)} = (\sum_{i=0}^{p-1} r_{(i)(i+1)} \cdot B, \sum_{i=0}^{p-1} P_{d_{(i)(i+1)}} + (\sum_{i=0}^{p-1} r_{(i)(i+1)}) \cdot k_0 \cdot B)$. In order to extract depth $P_{d_{(0)(p)}}$ of the traversed path, i.e., $\sum_{i=0}^{p-1} P_{d_{(i)(i+1)}}$, v_0 decrypts $E_{(0)(p)}$ with his/her own private key k_0. Then, v_0 decodes $P_{d_{(0)(p)}}$ to obtain the value $d_{(0)(p)}$. Finally, v_0 checks if this value satisfies the trust condition `TC`.

3.3.2 Trust computation

Literature offers several algorithms to compute the trust value between two indirectly connected nodes [93]. In general, all of them aggregate, in different ways, the trust values associated with edges (direct relationship) connecting the nodes in the path. In designing the trust token, we started by the assumption that trust values for direct relationships are associated directly by involved users. However, these values could be too much subjective or, even worse, users could not be able to significantly quantify the trustworthiness for a given contact, as they might have not enough information. To cope with this issue, in this chapter, we propose a trust computation able to automatically compute trust values between two adjacent nodes, The idea is that the resulting trust value can be considered as a suggested trust value between two users, which can then be multiplied with the user-given trust value, if present, as an *adjusting weight*. Among various trust algorithms, to compute the adjusting weight we selected the Tidal Trust algorithm [52]. This is known as the most famous and highly cited algorithm for inferring the trust between two adjacent nodes due to its simplicity and low complexity $O(V+E)$, which allows high scalability in its application. Thus, given two adjacent nodes v_i and v_j, the trust value associated with edge connecting v_i to v_j is $Tr_{ij} = \mu_{ij} \cdot Tu_{ij}$, where μ_{ij} is the adjusting weight automatically computed between them

according to the Tidal Trust algorithm, whereas Tu_{ij} is the trust value given by v_i to v_j, if any. Note that in case users are not able to assign a trust value, Tu_{ij} is set to 1. Then, given a path p, we compute the aggregated trust value as the average of values Tr associated with edges in p. To obtain this value, the trust token is generated and updated so as to contain the sum of Tr values of traversed edges. The payer can simply computes the average value by dividing this sum by the path depth.

In the following, we start presenting the privacy-preserving Tidal Trust computation. We then show how the obtained adjusting weights are used in the trust token computation.

Privacy-preserving Tidal Trust computation between two adjacent nodes. According to [52], the trust value for two adjacent nodes v_i and v_j is computed as follows:

$$\mu_{ij} = \frac{\sum_{s_t \in adj(i) \exists Tr_{(i)(s_t)} \geq max} \left(Tr_{(i)(s_t)} \cdot Tr_{(s_t)(j)}\right)}{\sum_{s_t \in adj(i) \exists Tr_{(i)(s_t)} \geq max} Tr_{(i)(s_t)}} \tag{3.1}$$

where $Tr_{(i)(s_t)}$, $Tr_{(s_t)(j)}$ denote the trust values, respectively, between v_i and v_{s_t}, and between v_{s_t} and v_j; $adj(i)$ is the adjacent node list of v_i; s_t is a common neighbor of v_i and v_j whose trust value satisfies the condition that $Tr_{(i)(s_t)}$ is greater than a given threshold max; t is the position of a common node s_t in the set $adj(i)$. In order to evaluate Formula 3.1, v_i and v_j have to execute the following steps:

(1) v_i and v_j collaboratively retrieve the set of common contacts, denoted as IS, by means of a privacy-preserving protocol. Here, we assume that such a protocol can be defined based on existing work (e.g., [45]). We plan to investigate such a protocol as a future work.
(2) For each common node s_t, v_j maps the trust value $Tr_{(s_t)(j)}$ between v_j and s_t to a point on (E) denoted as $P_{Tr_{(s_t)(j)}}$. Then, v_j encrypts $P_{Tr_{(s_t)(j)}}$ with the payer v_0's public key $(k_o \cdot B)$ and obtains $E_{(s_t)(j)} = (r_j \cdot B,\ P_{Tr_{(s_t)(j)}} + r_j \cdot k_0 \cdot B)$, where r_j is a random number generated by v_j. v_j also encrypts the point on (E) corresponding to s_t's identifier, denoted by $P_{ID_{s_t}}$ with v_i's public key. The resulting encrypted value is denoted by $E_{(j)}(P_{ID_{s_t}})$. All these values are sent to v_i as a set of pairs $[E_{(j)}(P_{ID_{s_t}}), E_{(s_t)(j)}]$, for each $s_t \in IS$.
(3) Once v_i receives pairs $[E_j(P_{ID_{s_t}}), E_{(s_t)(j)}]$, it decrypts the values $E_{(j)}(P_{ID_{s_t}})$, and decodes $P_{ID_{s_t}}$ to obtain identifiers of common nodes, i.e., ID_{s_t}. Among the resulting identifiers, v_i selects only those of the common nodes having $Tr_{(i)(s_t)} \geq max$, where max is a lower bound for trust value as in Formula 3.1. Let $\theta_{(i)(j)}$ denotes the set of common nodes satisfying this condition.
4) According to Formula 3.1, for each $s_t \in \theta_{(i)(j)}$, v_i has to compute $Tr_{(i)(s_t)} \cdot Tr_{(s_t)(j)}$. However, to apply the binary ECC, $Tr_{(s_t)(j)}$ is mapped to a corresponding point on (E), i.e., $P_{Tr_{(s_t)(j)}}$. Thanks to the additive property of the function used for mapping data in

point on (E),[3], we can calculate $Tr_{(i)(s_t)} \cdot P_{Tr_{(s_t)(j)}}$ instead. Moreover, since $P_{Tr_{(s_t)(j)}}$ is encrypted by v_0's public key in $E_{(s_t)(j)}$, it is possible to exploit the additive property of ECC. Thus, to compute $Tr_{(i)(s_t)} \cdot E_{(s_t)(j)}$, v_i adds to $E_{(s_t)(j)}$ the same value $E_{(s_t)(j)}$ for $(Tr_{(i)(s_t)} - 1)$ times.[4] We obtain a value, denoted as $addend_{s_t}$, such that $addend_{s_t} = Tr_{(i)(s_t)} \cdot E_{(s_t)(j)} = \sum_1^{Tr_{(i)(s_t)}} E_{(s_t)(j)} = (\sum_1^{Tr_{(i)(s_t)}} (r_j \cdot B), \sum_1^{Tr_{(i)(s_t)}} (P_{Tr_{(s_t)(j)}} + r_j \cdot k_0 \cdot B)) = (Tr_{(i)(s_t)} \cdot r_j \cdot B, Tr_{(i)(s_t)} \cdot P_{Tr_{(s_t)(j)}} + Tr_{(i)(s_t)} \cdot r_j \cdot k_0 \cdot B)$, for each $s_t \in \theta_{(i)(j)}$. Then, all these addends are summed together, to obtain an encrypted value of the numerator of the Formula 3.1, denoted by EN, where $EN = \sum_{t=1}^{|\theta_{(i)(j)}|} addend_{s_t} = ((\sum_{t=1}^{|\theta_{(i)(j)}|} Tr_{(i)(s_t)}) \cdot r_j \cdot B, \sum_{t=1}^{|\theta_{(i)(j)}|} (Tr_{(i)(s_t)} \cdot P_{Tr_{(s_t)(j)}}) + (\sum_{t=1}^{|\theta_{(i)(j)}|} Tr_{(i)(s_t)}) \cdot r_j \cdot k_0 \cdot B)$.

(5) v_i can easily compute denominator of Formula 1, denoted as $D = \sum_{t=1}^{|\theta_{(i)(j)}|} Tr_{(i)(s_t)}$. Hence, from the step 4 v_i has $EN = (D \cdot r_j \cdot B, \sum_{t=1}^{|\theta_{(i)(j)}|} (Tr_{(i)(s_t)} \cdot P_{Tr_{(s_t)(j)}}) + D \cdot r_j \cdot k_0 \cdot B)$. Then, it calculates the inverse value of the denominator, denoted D^{-1}, by a binary polynomial inversion calculation.

(6) Finally, v_i calculates the encrypted weights $E_{(i)(j)}(\mu_{(i)(j)}) = EN \cdot D^{-1}$, by adding EN to the same EN value for $(D^{-1} - 1)$ times according to the additive property of ECC, where $E_{(i)(j)}(\mu_{(i)(j)}) = (r_j \cdot B, \sum_{t=1}^{|\theta_{(i)(j)}|} (Tr_{(i)(s_t)} \cdot P_{Tr_{(s_t)(j)}}) \cdot D^{-1} + r_j \cdot k_0 \cdot B)$ After the above steps, the resulting encrypted value $E_{(i)(j)}(\mu_{(i)(j)})$ is used for the next step of computing the trust token.

b) Trust token. As discussed before, the trust token has to contain the summation of the trust values $Tr_{(i)(i+1)} = \mu_{(i)(i+1)} \cdot Tu_{(i)(i+1)}$ of each edge connecting v_i to v_{i+1} in the relationship path, for each $i \in \{0, p-1\}$. To preserve the privacy of participant nodes, each value $Tr_{(i)(i+1)}$ must be encrypted before being aggregated into the trust token. This can be easily done since the execution of the proposed privacy-preserving protocol for Tidal Trust algorithm between v_i and v_{i+1} already returns the encrypted value of $\mu_{(i)(i+1)}$, that is, $E_{(i)(i+1)}(\mu_{(i)(i+1)})$. Hence, we only need to add $E_{(i)(i+1)}(\mu_{(i)(i+1)})$ to $E_{(i)(i+1)}(\mu_{(i)(i+1)})$ values for $(Tu_{(i)(i+1)} - 1)$ times. Thus, according to the additive homomorphic property of ECC, we obtain the encrypted product $Tr_{(i)(i+1)}$, denoted by $E_{(i)(i+1)}$. Then, we can aggregate this product $E_{(i)(i+1)}$ into the trust token.

Trust token initialization. For each direct node to which the token will be forwarded, i.e., v_1, the payer v_0 first executes the privacy-preserving Tidal Trust computation to obtain $E_{(0)(1)}(\mu_{(0)(1)})$. Then, it computes $E_{(0)(1)} = E_{(0)(1)}(\mu_{(0)(1)}) \cdot Tu_{(0)(1)} = \sum_1^{Tu_{(0)(1)}} E_{(0)(1)}(\mu_{(0)(1)}) = E_{(0)(1)}(\mu_{(0)(1)} \cdot Tu_{(0)(1)}) = E_{(0)(1)}(Tr_{(0)(1)})$. We denote this encrypted value as $E_{(0)(1)} =$

[3]Let us denote a mapping function from a point P_{W_i} on (E) to a value W_i in F_2^m and vice versa as $P_{W_i} = f(W_i)$. We have $f(W_1) + f(W_2) = f(W_1 + W_2)$. In case $W_1 = W_2$, we have $f(W_1) + f(W_2) = f(2 \cdot W_1) = f(2 \cdot W_2)$.

[4]In support of this, in the execution of this part of the protocol, we assume that the trust value are mapped from $[0,1]$ to $[0,100]$.

$(r_{(0)(1)} \cdot B,\ \mu_{(0)(1)} \cdot Tr_{(0)(1)} + r_{(0)(1)} \cdot k_0 \cdot B)$, where $r_{(0)(1)}$ is a random number generated by v_1. Then, v_0 broadcasts $E_{(0)(1)}$ to its direct neighbors, i.e., v_1.

Trust token computation at intermediate nodes. Each intermediate node v_i receives from v_{i-1} the trust token. This contains $E_{(0)(i)}$, that is, the encrypted aggregation of trust values on the edges traversed so far. Similarly to the payer v_0, v_i executes the privacy-preserving Tidal Trust computation so as to obtain $E_{(i)(i+1)}(\mu_{(i)(i+1)})$, and then to compute $E_{(i)(i+1)} = E_{(i)(i+1)}(\mu_{(i)(i+1)}) \cdot Tu_{(i)(i+1)} = \sum_1^{Tu_{(i)(i+1)}} E_{(i)(i+1)}(\mu_{(i)(i+1)}) = E_{(i)(i+1)}(\mu_{(i)(i+1)} \cdot Tu_{(i)(i+1)})$. We denote this encrypted value as $E_{(i)(i+1)} = (r_{(i)(i+1)} \cdot B, \mu_{(i)(i+1)} \cdot Tr_{(i)(i+1)} + r_{(i)(i+1)} \cdot k_0 \cdot B)$, where $r_{(i)(i+1)}$ is a random number generated by v_{i+1}. Then, v_i computes $E_{(0)(i+1)}$, which is given as the sum between the received $E_{(0)(i)}$ and the just computed $E_{(i)(i+1)}$. It obtains:

$$\begin{aligned} E_{(0)(i+1)} &= E_{(0)(i)} + E_{(i)(i+1)} \\ &= (r_{(0)(i)} \cdot B + r_{(i)(i+1)} \cdot B,\ (\mu_{(0)(i)} \cdot Tr_{(0)(i)}) \\ &+ (\mu_{(i)(i+1)} \cdot Tr_{(i)(i+1)}) + r_{(0)(i)} \cdot k_0 \cdot B + r_{(i)(i+1)} \cdot k_0 \cdot B) \\ &= (\sum_{i=0}^{p-1} (r_{(i)(i+1)} \cdot B),\ \sum_{i=0}^{p-1} (\mu_{(i)(i+1)} \cdot Tr_{(i)(i+1)}) \\ &+ (\sum_{i=0}^{p-1} (r_{(i)(i+1)}) \cdot k_0 \cdot B)) \end{aligned} \tag{3.2}$$

Trust token validation. The propagation ends when the tokenset reaches the payee v_p. This tokenset then is sent back to the payer v_0 that decrypts the tokenset content, i.e., $E_{(0)(p)}$ with its private key (k_0). Thus, v_0 obtains the aggregated trust value $Tr_{(0)(p)} = \sum_{i=0}^{p-1} (\mu_{(i)(i+1)} \cdot Tr_{(i)(i+1)})$. As such, the payer v_0 obtains the final value as follows:

$$\begin{aligned} D(E_{(0)(p)}) &= D(\sum_{i=0}^{p-1} (r_{(i)(i+1)} \cdot B),\ \sum_{i=0}^{p-1} (\mu_{(i)(i+1)} \cdot Tr_{(i)(i+1)}) \\ &+ (\sum_{i=0}^{p-1} (r_{(i)(i+1)} \cdot k_0 \cdot B)) = \sum_{i=0}^{p-1} (\mu_{(i)(i+1)} \cdot Tr_{(i)(i+1)}) \\ &+ \sum_{i=0}^{p-1} (r_{(i)(i+1)}) \cdot k_0 \cdot B) - \sum_{i=0}^{p-1} (r_{(i)(i+1)} \cdot B \cdot k_0) \end{aligned} \tag{3.3}$$

$$Tr_{(0)(p)} = \sum_{i=0}^{p-1} (\mu_{(i)(i+1)} \cdot Tr_{(i)(i+1)}) \tag{3.4}$$

The obtained $Tr_{(0)(p)}$ is still a point on (E), v_0 needs to decode it to a value in F_2^m.

3.3.3 Relationship type computation

We assume that the payer v_0 initializes the relationship type token, hereafter *rt* token, which is then forwarded only to those v_0's neighbors with which v_0 has a relationship type equal to rt_{TC}. The token is initialized as the encryption of the sum of points P_τ and $P_{rt_{(0)(1)}}$ on (E), respectively, corresponding to a random value $\tau \in F_2^m$ and the relationship type $rt_{(0)(1)}$ that is, the type in rt_{TC}. Each intermediate node v_i adds to the content of the received *rt* token the encrypted point $P_{rt_{(i)(i+1)}}$ on (E) corresponding to the relationship type of the edge to be traversed, that is, $e = (v_i, v_{i+1})$. This is done until the *rt* token reaches the payee v_p that then sends the token back to the payer v_0. In turn, the payer v_0 decrypts *rt* token content and verifies if *rt* is compliant with TC. A detailed description is presented as follows.

Relationship type token initialization. The payer v_0 sets a random value $\tau \in F_2^m$ and encodes τ to a point P_τ on (E). v_0 also encodes the relationship type $rt_{(0)(1)}$, which is also the one required in rt_{TC}, to a point on (E) denoted by $P_{rt_{(0)(1)}}$. Then, v_0 uses its public key $(k_0 \cdot B)$ to obtain the encrypted value $E_{(0)(1)}$. As such, $E_{(0)(1)} = (Y1_{v_0}, Y2_{v_0}) = (r_{(0)(1)} \cdot B,\ (P_\tau + P_{rt_{(0)(1)}}) + r_{(0)(1)} \cdot k_0 \cdot B)$, where $r_{(0)(1)}$ is a random integer generated by v_0. After that, v_0 sends $E_{(0)(1)}$ to each of its direct neighbor, i.e., v_1, with which it has a relationship of type rt_{TC}.

Relationship type token computation at intermediate nodes. Once the intermediate node v_i receives the *rt* token from v_{i-1}, it encodes $rt_{(i)(i+1)}$ to a point $P_{rt_{(i)(i+1)}}$ on (E). Then, v_i computes the encrypted value of $P_{rt_{(i)(i+1)}}$, that is, $E_{(i)(i+1)} = (Y1_{v_i},\ Y2_{v_i}) = (r_{(i)(i+1)} \cdot B,\ P_{rt_{(i)(i+1)}} + r_{(i)(i+1)} \cdot k_0 \cdot B)$, where $r_{(i)(i+1)}$ is a random integer generated by v_i. Then, v_i aggregates $E_{(i)(i+1)}$ into the *rt* token content, i.e., $E_{(0)(i)}$, and obtains $E_{(0)(i+1)} = E_{(0)(i)} + E_{(i)(i+1)} = ((r_{(0)(i)} + r_{(i)(i+1)}) \cdot B,\ (P_\tau + P_{rt_{(0)(i)}} + P_{rt_{(i)(i+1)}}) + (r_{(0)(i)} + r_{(i)(i+1)}) \cdot k_0 \cdot B)$, where $P_{rt_{(0)(i)}} = \sum_{l=0}^{l=i-1} P_{rt_{(l)(l+1)}}$ is an aggregated point on (E) corresponding to the aggregated relationship types on the edges $e = (v_{(l)}, v_{(l+1)})$ traversed from v_0 to v_i. Then, v_i sends $E_{(0)(i+1)}$ to v_{i+1}. The propagation is iterated until the *rt* token reaches the payee v_p, then v_p sends the *rt* token back to the payer v_0.

Relationship type token validation. Once the payer v_0 receives $E_{(0)(p)}$ from the payee v_p, v_0 decrypts $E_{(0)(p)}$ with its own private key k_0 and obtains $P_\tau + P_{rt_{(0)(p)}} = P_\tau + \sum_{i=0}^{p-1} P_{rt_{(i)(i+1)}}$. Then, v_0 subtracts the decrypted value to P_τ and gets $P_{rt_{(0)(p)}}$.

Finally, if all types on the relationship path are equal, it means that: $rt_{(0)(p)} = \sum_{i=0}^{p-1} rt_{TC} = \lambda \cdot rt_{TC}$, where λ is the number of hops to reach the payee from the payer or the depth through which the *rt* token propagates. We have the above equation due to the basic point scalar multiplication operator on (E). Then, v_0 divides $rt_{(0)(p)}$ by rt_{TC}, by a binary polynomial division. If the division does not result in a remainder and the quotient is equal to the relationship depth extracted from the depth token, it means that all edges in the traversed path

have the type equal to rt_{TC}. Otherwise, at least an edge has $rt \neq rt_{TC}$, so the trust condition is not satisfied.

3.4 Flooding Optimization

The path finding protocol described so far assumes that each intermediate node sends the tokenset to each of its direct contacts. However, that broadcast approach might not be suitable in a mobile network. Although the tokenset computation does not require plenty of resources (see experiments in Section 3.6.2), the broadcast approach might imply a high bandwidth usage since the number of tokensets that are exchanged rapidly increases (see experiments in Section 3.6.1). In order to cope with this issue, we introduce an optimization method aiming at reducing the number of propagated tokensets. An easy way for drastically reducing the number of tokensets is to let the intermediate nodes be aware of which type of relationship the payee is looking for. Given this information, they would be able to forward the tokenset only to the contacts with which they have the required relationship, rather than to broadcast to all their neighbors. However, this solution would reveal the required relationship type. In order to restrict the number of messages by at the same time protecting the required relationship type, we adopt a solution inspired by k-anonymity [137]. The basic idea is to let the intermediate nodes know a set, denoted by A_{rel}, of k relationship types, $k-1$ of which are chosen randomly according to a uniform distribution from RT whereas one is that required in `TC`. The flooding is then limited only to those edges labeled with a relationship type among the k types in A_{rel}.

The rt token is computed and validated as described in Section 3.3.3. Thus, when the payer v_0 obtains the final aggregated relationship type, it decrypts the value and verifies if `TC` is satisfied. Similarly to k-anonymity, the inference of the real relationship type required by a trust preference depends on the selected k value. This value can be selected based on the domain requirements, trying to trade off between efficiency and privacy of trust preferences, since the smaller is A_{rel}, the better is the protocol performance, but the greater is the possibility to infer the required relationship type.

3.5 Security properties

In this section we discuss the security properties ensured by the proposed protocol. As stated in Section 1, we are interested in protecting relationship information. In what follows, we describe which information can be inferred by nodes involved in the protocol. This discussion is done under the assumption that nodes are *honest but curios*, as such, they all perform steps

required by the protocol, by trying to gain reserved information, if possible. The discussion is organized according to the node's role.

Intermediate nodes. We recall that intermediated nodes receive tokens containing encrypted aggregated information of path traversed so fa. Then, by exploiting *ECC*, they add the properties (i.e., trust, depth, relationship type) of the edge through which the token is going to be sent. According to token definitions, only v_0 is able to decrypt the aggregated path information, since all tokens have been encrypted with v_0's public key. Moreover, *ECC* keys used in *SmartPay* have sizes larger than 160 bits [87], which meets the security conditions in cryptanalysis [72]. Additionally, the fact that a relationship path includes many of nodes implying the combination of the random big numbers makes adversaries hard to run a Bruce Force algorithm to analyze the tokens. This, even for the depth token that has alway value '1', since different nodes give different encrypted values with different random big numbers.

Payee. This node has only to send back to the payer the received token. Similar to intermediate nodes, it is not able to access aggregated data. According to honest-but-curios model, this node can neither generate nor add fake tokens to the received one, before sending it to payer.[5]

Payer. This is the only node able to decrypt aggregated data, as such obtaining the depth, type and trust of the traversed path p. These are all aggregated data, that do not reveal nodes that are involved in p, as well as, property of each edge. However, there is a particular case, where from the aggregated data payer can infer private information. This is the case of paths of depth 2. Indeed, in this case the path has a unique intermediate node, say X, which is also common friend of both payer v_0 and payee v_r. Thus, X has an incoming edge from v_o, and an outgoing edge to v_r. v_0, once decrypted the token, is aware of the depth, that is, it is aware that one of its contacts has a direct relationship with v_r. It also knows that it is one to which the token has been sent, thus, the one with which v_0 has a relationship of rt_{TC} type. However, it does not know exactly who, but it can guess with a probability $P(X) = \frac{1}{\|NC\|}$, where $\|NC\|$ is the number of direct contacts of v_o with rt_{TC} type. Larger is $\|NC\|$, smaller is the ability of v_0 to infer information about X. We have to note that, even if this is possible, this is not likely to happen as, in general, social network users have several hundreds of friends. As an example, on average, Facebook users have 130 friends. If we consider that SmatPay merges different user social network accounts, we believe that $\|NC\|$ ensures a low probability of guessing who is X.

[5] It is important to note that even in case of not-honest behavior, adding fake edge properties will decrease the possibility of satisfying the trust condition TC. For example, dding a depth token increases the depth making thus more difficult to satisfy the length limit stated in $depth_{TC}$.

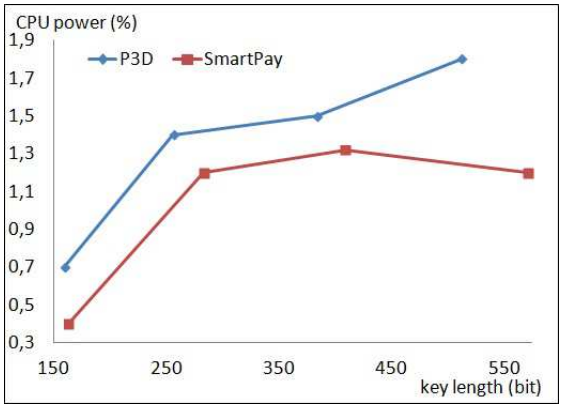


Fig. 3.2 CPU power consumption for creating a token on a node (%)

3.6 Experiments

We carried out several experiments to test the proposed protocol. In particular, the first experiment illustrates the cost in terms of CPU, memory and time of token computation. Then, we present the performance of the optimizing flooding strategy introduced in Section 3.6.2. The physical resource used for the experiments is a PC configured with Core2 Duo CPU 3GHz, 4.0GB DDRAM, 64 bit Windows 7 Professional.

3.6.1 Computational costs

In this section, we compare the costs of the proposed protocol with another decentralized path finding protocol, that is, P^3D [156] due to its similarity with our protocol. P^3D is a fully decentralized protocol for the social networks based on homomorphic cryptography has been proposed, solving the above mentioned problems. However, the main difference between SmartPay and P^3D is that we make use of light cryptography techniques. As such the proposed protocol can easily be executed by mobile devices due to its more effective consumption of system resources, and faster performance.

Technically, in order to aggregate path properties, P^3D uses the cryptographic hash function $SHA-1,2$ combined with either the property of logarithm function or Elgamal cryptographic algorithm. According to the experimental results of P^3D, the logarithm function is much faster than Elgamal. Therefore, we make a comparison between *SmartPay* and P^3D variant based on $SHA-1,2$ and logarithm function. The key sizes of $SHA-1,2$ are sequentially 160, 256, 384, 512 bits. In *SmartPay*, we use *Koblitz elliptic curves* (or,

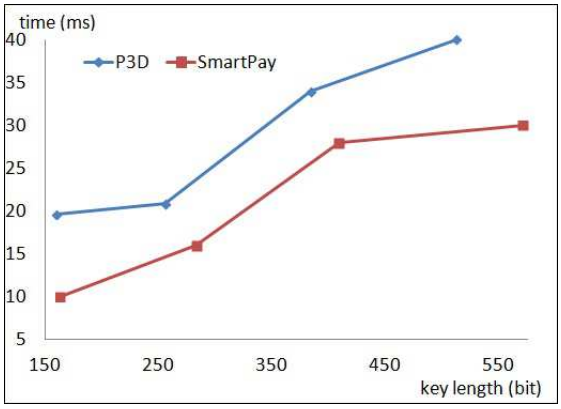


Fig. 3.3 Time consumption for creating a token on a node (ms)

anomalous binary curves), which are one among the 15 recommended elliptic curves analyzed in [59]. These elliptic curves are based on the binary F_2^m and the size of underlying fields are 163, 283, 409, and 571 bits in turn. (See more details in [59], [79], [80], [42]). Our first experiment consists in computing the tokenset, i.e., depth, trust and rt token, tracing the system resources usage, i.e., CPU, memory, and time. The tokenset has been computed 10 times. The final resources usage is then given as the average of these 10 executions.

Figures 3.2, 3.3, and 3.4 show the comparison in terms of CPU power, memory, and time usage between *SmartPay* and P^3D. From experimental results, we can easily see that resource usage of *SmartPay* is less than P^3D. With the largest key size of *SmartPay* (571 bits) and P^3D (512 bits), P^3D used 1.8% CPU power w.r.t. 1.2% of *SmartPay* (see Figure 3.2). Meanwhile, the space for P^3D is 4640 bytes while *SmartPay* uses 728 bytes (see Figure 3.4). *SmartPay* consumes 6.3 times less than P^3D, which is an interesting property considering the resource-constrained mobile domain. Moreover, the time spent on creating the tokenset with P^3D is 40ms, 1.3 times more than *SmartPay* that requires only 30 ms (see Figure 3.3). Let us consider the smallest key size of *SmartPay* (163 bits) and P^3D (160 bits): P^3D used 0.7% CPU power while *SmartPay* used 0.4%, 1.75 times more than *SmartPay* (see Figure 3.2). Meanwhile, the space for P^3D is 3120 bytes, while *SmartPay* uses 536 bytes (see Figure 3.4). P^3D consumes 5.8 times more than *SmartPay*. Time spent on creating a token with P^3D is 19.6ms, 1.96 times more than *SmartPay*, i.e., 10ms (see Figure 3.3). With other key sizes, P^3D also consumes resource much more than *SmartPay*. Consequently, we conclude that *SmartPay* is more efficient than P^3D, especially in mobile environment.

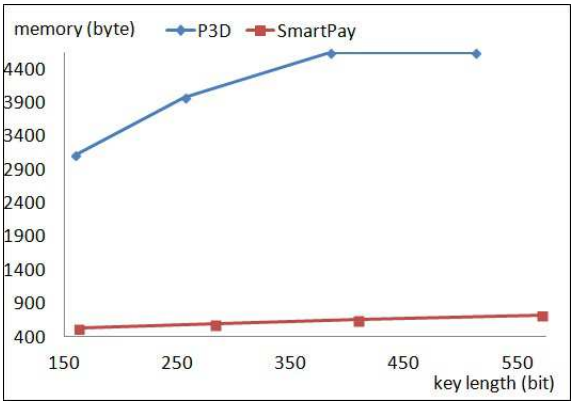


Fig. 3.4 Space load for creating a token on a node (byte)

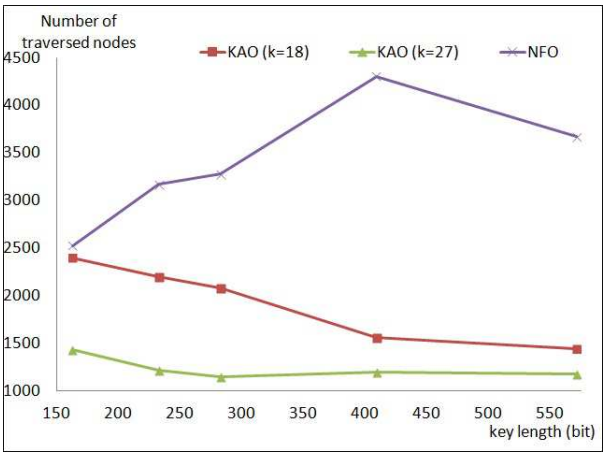


Fig. 3.5 A comparison between NFO and KAO about number of traversed nodes per second

3.6.2 Flooding optimization

In order to verify the optimizating flooding strategy introduced in Section 3.4, we make experiments on an existing social network dataset. In particular, we used the Epinions, dataset[6] which has been extracted from a Who-trusts-whom online social network.[7]

This is a dataset representing a directed social graph with 75,879 nodes and 508,837 edges. In order to perform the experiments, we modified the dataset so as to associate with each edge a relationship type and a trust value. The relationship type is uniformly and randomly picked up from [32] based on the ontological FOAF vocabulary specification

[6]http://snap.stanford.edu/data/
[7]http://www.epinions.com/

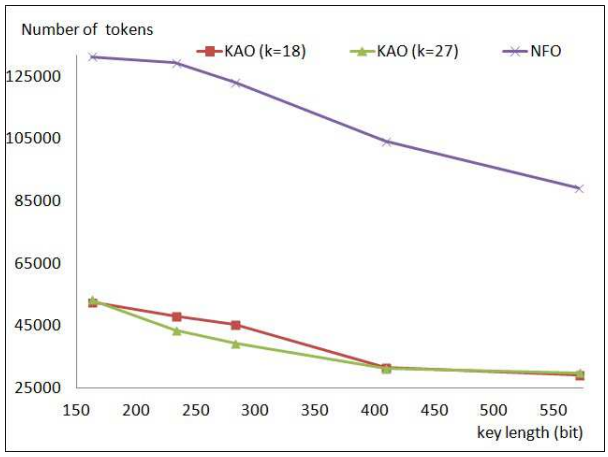


Fig. 3.6 A comparison between NFO and KAO about number of flooding tokens per second

which provides a set of 35 relationship types. A trust value set on an edge $e \in E$ is also randomly generated uniformly from a range $[0,1]$.

In order to measure the effects of the flooding optimization, we execute the protocols several times by tracing: the *number of traversed nodes* and the *number of generated tokens*. We execute the protocol according to the broadcast technique, i.e., *no flooding optimization - NFO* and the *k-anonymity based optimization - KAO* with different k values. In particular, in choosing the value k we have considered that if this k is too large, the expected KAO performance is close to the one of NFO. In contrast, if k is too small, the relationship type can be easily inferred. To set a good trade-off we have selected $k = 18$ and $k = 27$, that is half and $3/4$ of available relationship types in [32]. Besides, we use the key sizes of 163 bits, 233 bits, 283 bits, 409 bits and 571 bits. Figures 3.5, 3.6 present the comparison of the flooding methods in which the x axis represents the key size of *ECC* algorithm; and the y axis reports the number of traversed nodes in Figure 3.5, and the number of tokens flooding in the network in Figure 3.6.

In case of the smallest 163 bit key size, in NFO, the number of traversed nodes is 2522 whereas the number of relevant tokens is 131313. In contrast, KAO with $k = 18$ traverses 2393 nodes (1.05 times less than NFO) and 52377 relevant tokens (2.51 times less than NFO), with $k = 27$ traverses 1424 nodes (1.77 times less than NFO) and 53230 relevant tokens (2.47 times less than NFO). Now let us consider the largest key size (571 bits), NFO generates 89139 tokens flooding through 3669 nodes. Meanwhile, KAO ($k = 18$) generates 28972 tokens (3.08 times less than NFO) going through 1437 nodes (2.55 times less than NFO), while KAO ($k = 27$) method generates 29799 tokens (2.99 times less than NFO) going

through 1175 nodes (3.12 times less than NFO). With the other key sizes, the experimental results also prove that KAO is completely effective.

3.7 Complexity Analysis

In order to assess the efficiency of the protocol more exactly, we consider the time complexity of the fact that a token moves from the payer through intermediate nodes to the payee. We denote the time consumption of processing a token at one node as t_p. t_p is formulated as a summation of the token updating time (i.e., t_u), the transmission time (i.e., t_t), and the waiting time on sensing the connection signal from the node's contacts (i.e., t_w). More formally:

$$t_p = t_u + t_t + t_w$$

Notice that t_u depends on a couple of factors, that is, the physical power of the mobile device and the key size of used cryptographic algorithms as mentioned in Section 3.6. Whereas, t_w is '0' if there are nodes connecting the sending node and available to receive the token.

Since the time complexity of the path discovery depends on the number of active nodes, we assume that at time t, there are N nodes being on in the network. We consider two cases in which the path discovery can be done in the farthest and slowest way. In the case that the two nodes are adjacent, the number of intermediate nodes is "1". The time cost in the above equation is still applicable to this case as the sending node needs to compute the token then to transfer the updated token to the next node. Hence, the time complexity is $O(N) = 1 \times t_p = t = (t_u + t_t + t_w)$. In the worst case, the token needs to move through $(N-1)$ nodes, excluding the payee as the payee does not process the token, before reaching the payee, the the time complexity turns out to be $O(N) = (N-1) \times t_p = (N-1) \times (t_u + t_t + t_w)$.

Chapter 4

Secure Mobile Person-To-Person Payment over Mobile Ad-hoc NETwork

In Chapter 3, we have presented a secure protocol, namely SmartPay [22, 23], to exploit the available social network connections to support users in making decisions behind mobile P2P payments. In particular, SmartPay leveraged social connections to help the payers, i.e., those people who give credit to someone, to judge if the person asking money can be considered trusted. More precisely, SmartPay exploits social networks to verify how payer and payee are connected in the social graph. Based on this information and the payer trust preferences, SmartPay suggests whether the money transfer should be authorized or not. A key aspect of SmartPay is the decentralized protocol exploited to gather information on the social path connecting the payer and the payee, which is then encoded into a data structure, called *tokenset*. Furthermore, SmartPay exploits a light cryptographic algorithm, namely binary Elliptic Curve Cryptography (ECC) algorithm [71] to protect the privacy of user relationship information in the traversed path. This makes SmartPay able to privately aggregate information on the traversed paths (i.e., depth, trust, relationship type), without revealing any information on traversed edges. However, in [22, 23] we did not cope with the communication means for exchanging tokensets among SmartPay users. It is only assumed that the communication means is Internet (e.g., using emails or sockets). In this chapter, we want to explore alternative communication means. In particular, we propose to exploit Mobile Ad-hoc NETwork (MANET). Several applications have been deployed over MANET (see Chapter 1, Section 1.1.1 for examples). Plenty of MANET applications have been deployed successfully, and particularly for scenarios where local networks with a high density of users are available (e.g., a conference room, library, supermarket, stadium, park, university campus, company buildings). This typical MANET scenario fits well the one in which P2P mobile payment applications can be used. Hence, we strongly believe that there is a need of

deploying P2P payments over MANET. For this reason, in this chapter, we investigate how to deploy SmartPay over MANET. To our knowledge, this work is the first one exploiting social network relationships for P2P payments over MANET.

Despite the benefits, this new communication means has open challenges that need to be addressed [126] [57]. Among them, the most relevant are: *(1) privacy and security*, *(2) energy efficiency*, and *(3) network performance*. The deployment of SmartPay over MANET has to cope with all these issues. Regarding privacy and security, the exchange of tokensets through MANET rises new challenges w.r.t. of those investigated in [22, 23]. Indeed, according to the MANET protocol, a tokenset might be forwarded through several mobile users before reaching the destination. Those users are not involved in the path traversal and thus do not have to infer any information about the payment transaction as well as aggregate relationship information. In addition, in [22, 23] the payer identity was not kept private, as the proposed solution exploits the payer's public key. In contrast, in this chapter we present a solution suitable to MANET communication that provides a more secure protocol than the one in [22, 23]. Regarding issue (2), we use cryptographic algorithms on elliptic curves, that is, binary ECC [71] and Elliptic Curve Digital Signature Algorithm (ECDSA) [69]. These binary cryptographic algorithms make the mobile devices able to perform efficient computations and consume less energy. With respect to (3), SmartPay adopts a strategy inspired by k-anonymity to optimize network performance. However, selecting a reasonable value of k to make the network consumption effective is not easy. In particular, in case k is large, network performance is similar to the one of broadcasting techniques, but privacy is preserved more strongly. In case k is small, the bandwidth consumption is low but privacy cannot be preserved effectively. To overcome this problem, in this chapter, we propose an optimization strategy exploiting secure comparison algorithms so as to reduce the flooding of tokensets in MANET. In particular, our algorithm allows to evaluate encrypted relationship information inside a tokenset to be forwarded in the network.

The remainder of the chapter is organized as follows. A short MANET introduction is introduced in Section 4.1. Section 4.2 presents issues and possible solutions to deploy SmartPay over MANET. Section 4.3 describes the expanded SmartPay protocol over MANET. Section 4.4 presents the optimization strategy. Experimental results are reported in Section 4.5. Finally, security properties are discussed in Section 4.7.

4.1 MANET (Mobile Ad-hoc NETwork) introduction

MANET is a collection of many devices equipped with wireless communications and networking capabilities that make them able to communicate with each other [143]. This can

be done through a direct communication, if the destination of the message is within the *radio range* of the sender. In general, the radio range depends on the communication means. For instance, a Bluetooth device has a radio range of 10m, whereas a Wi-Fi device has a radio range of 100m [44]. In case two devices are out of their radio ranges, a protocol is initialized aiming at forwarding the message through intermediate nodes until reaching the final destination. For this purpose, each device, hereafter *node*, keeps track of the set of neighbors that fall into its radio range. Since nodes might leave or enter the other's radio range, the MANET topology can change in a very dynamic way. These changes might split the network into different subnetworks, called partitions, among which there might not exist a possible routing path. To support the communication among nodes in different partitions, large-scale MANETs make use of connectors. These are proxies responsible for forwarding messages from a partition to another. A connector can be a fixed device (such as an Access Point (AP), a server, or a switch that functions as a gateway) [154], or mobile nodes/robots [37]. When a node joins the network, it receives from its neighbors information about available connectors. When a node wants to transmit a message to a node in another partition, it forwards the message to connectors in its connectors list.

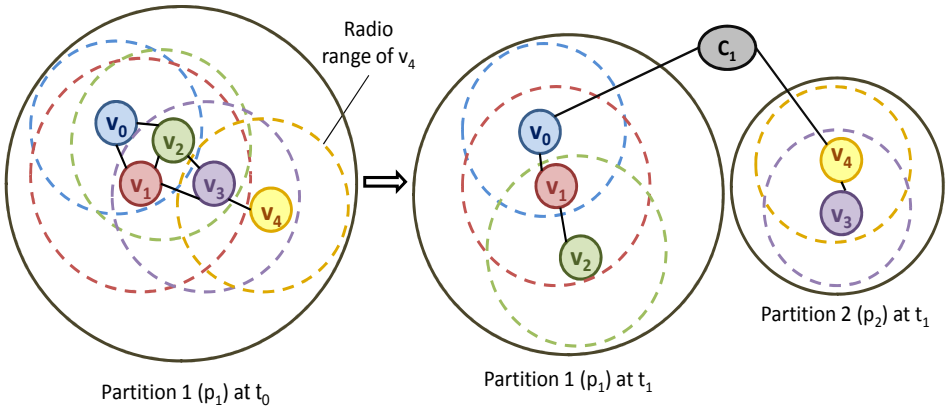


Fig. 4.1 MANET partitions and radio ranges at different times

Example 1. Figure 4.1 illustrates a MANET at two different time instants, i.e., t_0 and t_1. Each node has a radio range, denoted by the surrounding dotted circle. Partitions are denoted by a solid circle. When a node enters in another dotted circle, the two nodes can communicate directly using Wi-Fi standards, i.e., IEEE 802.11. At time t_0, v_0 can directly contact v_1 since it is in v_1's radio range and vice versa. This direct communication is reported as an edge connecting v_0 and v_1, denoted as $e(v_0, v_1)$. In contrast, v_0 can contact v_3 indirectly by a protocol forwarding the message through a routing path, e.g., $v_0 \rightarrow v_2 \rightarrow v_3$. As reported in

Figure 4.1, at time t_0 all mobile nodes v_0, v_1, v_2, v_3, v_4 set up a partition, say p_1, since all of them can communicate with each other directly or indirectly. Figure 4.1 reports that v_3 and v_4 leave p_1 at time t_1 and form the second partition, say p_2. The two partitions, i.e., p_1, p_2, communicate with each other through the connector, that is, C_1.

4.2 SmartPay over MANET: Issues and Solutions

To be enforced in MANET, SmartPay needs to be expanded so as to handle MANET communications. In order to distinguish users in SmartPay social network and in MANET, we organize the structure of Smartpay over MANET into two layers, that is, SmartPay Social Network Layer (SSNL) and Smartpay MANET Layer (SMNL). Users in the SSNL are called contacts, whereas users in the SMNL are called nodes. Hereafter, let v_i^S denote a contact i, and v_i^M denote a node i. One contact in the SSNL is mapped onto one node in the SMNL, and vice versa. SSNL includes contacts operating according to the SmartPay protocol. In particular, the SSNL is responsible for aggregating path information and propagating the tokenset to the other contacts. Whereas, the SMNL includes nodes (physical mobile devices) and communications in MANET. MANET communications are set up according to requests on SSNL. The SMNL is in charge of forwarding the tokenset to neighbors, or stopping the tokenset when the destination is reached. It is important to note that while contacts in SSNL are mapped on nodes in SMNL, the same does not hold for edges in the corresponding graphs. Indeed, in a MANET graph, edges are defined based on nodes' radio ranges, and not based on relationship information like in SSNL. As such, the protocol for discovering a path in a social graph defined in [22, 23] cannot be applied as it is in the MANET graph. Indeed, the tokenset flows might follow a different path in the MANET layer, due to the availability and the position of nodes at that moment, as the following example clarifies.

Example 2. Let us consider the scenario of a MANET created among mobile devices of spectators of a football match at a big stadium. Let us assume some of them are connected in a social network graph. As presented in Figure 4.2, this MANET consists of several partitions, that are connected together by means of a set of connectors (wireless access points) placed around the stadium. Let us consider four friends: Bob, Evans, Carl, and Alice, that come to the stadium to follow the match, but they sit in different stadium areas. Assume that Bob is sitting next to Evans, Carl is a little far from Evans and Bob, while Alice is at the farthest place from the others. Based on the distance, they are located into two different partitions (see Figure 4.2). Evans and Bob can communicate directly because they are in the radio range of each other. If Bob wants to communicate with Carl, he must communicate through two intermediate nodes, that is, Evans and Davis. If Alice wants to communicate

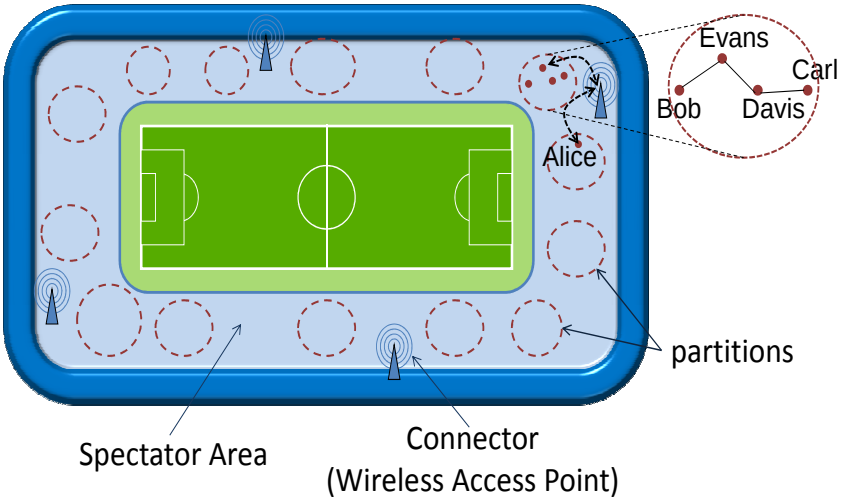


Fig. 4.2 Communication at a stadium

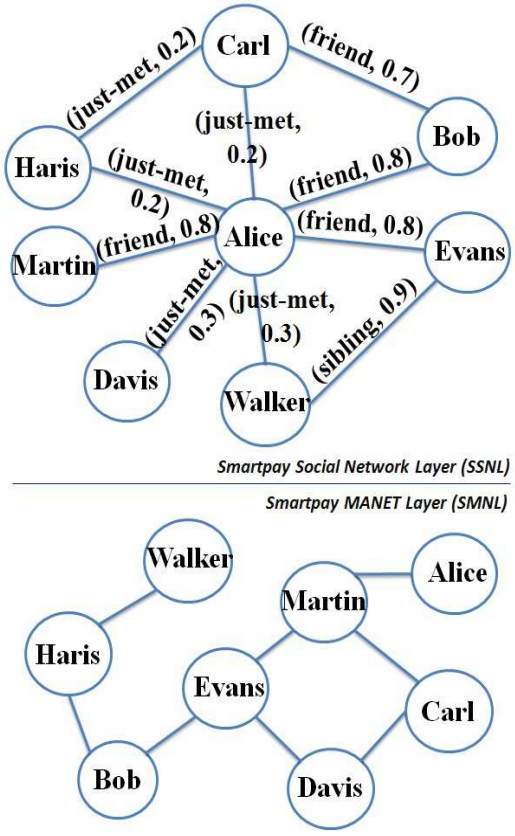


Fig. 4.3 SmartPay protocol over MANET

with Bob, her message needs to be sent to a connector that then forwards the message to Bob's partition. Continuing with this scenario, let us consider the social graph and MANET described in Figure 4.3. As depicted in Figure 4.3, connections in SSNL and SMNL are different. Suppose that Bob wants to buy snack and beverage, and thus he asks Evans for lending him money by making a direct request. Evans and Bob are not connected in SSNL, Evans does not know Bob very well, so he initializes and sends a SmartPay tokenset to Bob through the SmartPay application installed on his cell phone. The tokenset propagates from Evans to Bob through SSNL path Evans→Alice→Bob. Whereas, in the SMNL the tokensets go directly to Bob through the path Evans→Bob.

Due to the SSNL communication layer, the SmartPay protocol as described in [22, 23] cannot be directly applied, as further information needs to be inserted in the tokenset in order to route it in the MANET network. As it will be described in the next section, the revised tokenset has still to ensure privacy and security properties.

4.3 Expanded Smartpay Protocol (ESP)

Let us consider the three main roles with which nodes participate in the protocol, that is, the payer, the payee, and the intermediate nodes.

The Payee. Let us assume that a user v_{payee} wants to borrow an amount of money from another user, say v_{payer}. He/she first needs to send v_{payer} a request. By using SmartPay, the payer will generate a tokenset which is propagated in the social graph until it reaches the payee. The tokenset has to contain some information that makes the payee able to determine he/she is the final destination of the received tokenset or this relates to another execution of ESP.

The naive idea is to insert the payee identity in the tokenset. Yet, the payee identity is a personal information. Thus, we use an additional information, called *Validator*, which is generated as the encryption with the public key of the payee, i.e., $(K_{payee} \cdot B)$, of some local information gathered from the payee's mobile device. These are, as an example, the MAC address and the timestamp of the instant when the payee makes a request to the payer. The timestamp is used as a session identity to distinguish the payment request at the payee side, so that the payee can recognize exactly the request relating to the payer. Before encrypting the combination of MAC address and timestamp, to improve the robustness of Validator against adversaries (see Section 4.7), we take the combination of MAC address and timestamp apart into groups of bytes, then permute randomly positions of these groups of bytes by applying the Fisher-Yates shuffle algorithm [41]. Note that the random arguments used for permutation are held by the payee, so that the payee can reuse them for its identity data recovery. After

that, the payee encrypts the permuted combination of local information by its public key to gain Validator. Then, Validator is appended at the end of the ESP tokenset. Also notice that Validator is not modified through the path from the payer to the payee.

Definition 1. *(Validator).* Let v^M_{payee} be the node who makes the payment request, $MA_{v^M_{payee}}$ be the MAC address of v^M_{payee}, $t_{v^M_{payee}}$ be the timestamp when v^M_{payee} makes the request, $(K_{v^M_{payee}} \cdot B)$ be the public key of v^M_{payee}. The Validator made by v^M_{payee} is defined as follows:

$$Validator_{v^M_{payee}} = Enc_{(K_{v^M_{payee}} \cdot B)}(shuffle([MA_{v^M_{payee}} || t_{v^M_{payee}}]))$$

where shuffle() is the Fisher-Yates shuffle algorithm, '||' is an operation concatenating two bit strings.

v_{payee} inserts the Validator into the request and sends the request to v_{payer}.

Example 3. Let us continue with Example 2 (see Figure 4.3), and suppose once again that Bob wants to borrow money from Evans. Therefore, he needs to generate the request along with the Validator and sends them to Evans. Bob first retrieves the request timestamp, i.e., t_{Bob^M}, and the MAC address, i.e., MA_{Bob^M}, from his mobile device. Then, he combines them together, creates a random permutation on the obtained combination, and then encrypts the resulted permutation with his public key to generate the Validator: $Validator_{Bob^M} = Enc_{(K_{Bob^M} \cdot B)}(shuffle[MA_{Bob^M} || t_{Bob^M}])$. Bob sends then the Validator and the payment request to Evans.

After that, v_{payee} waits for an ESP tokenset from v_{payer} through his/her contacts. When v_{payee} receives an ESP tokenset including his/her Validator, he/she sends that ESP tokenset back to the payer and waits for the payment result.

Example 4. Consider Example 3, when Bob receives a tokenset with the Validator $Enc_{(K_{Bob^M} \cdot B)}$ $(shuffle([MA_{Bob^M} || t_{Bob^M}]))$, he can use his private key (i.e., K_{Bob^M}) to successfully decrypt the Validator, and verifies that he is the destination of the tokenset. Then, Bob stops forwarding the tokenset to his neighbors.

The Payer. In defining ESP we wish to cope with an open issue in [22, 23]. Indeed, in order to aggregate information into the tokenset, in [22, 23] we assumed that each user knows the public keys of other participants. However, this makes users involved in the SmartPay execution able to infer who is the payer. In ESP, we wish to protect this information as well. Therefore, we assume that the payer, i.e., v^S_{payer}, generates a pair of temporary ECC public and private keys, i.e., $(K_{tempv^S_{payer}} \cdot B)$ and $K_{tempv^S_{payer}}$, for each new transaction, to be used instead of its original public key so that intermediate nodes cannot infer the payer identity.

Temporary keys have to be protected so as to avoid other contacts to misuse them. Beside, in order to avoid that intermediate nodes do aggregation on the same tokenset received from the same SSNL node in case the sending node can be subject of a replay attack (see Section 4.7), we insert the information of session identity (ID) between two SSNL nodes, called *sessionID*. *sessionID* can include an ID of the receiving node and a number standing for the session order. Beside, to let the receiving nodes know the sending node, we insert the ID of the sending node, namely $ID(v_i)$, into the tokenset. Both $ID(v_i)$ and *sessionID* should be protected as the temporary public key of the payer. At this purpose, we require that the temporary public key, the *sessionID* and $ID(v_i)$ are encrypted by the intermediate contacts who have used it to aggregate the relationship information with the official public key of the SSNL contact to which the tokenset has to be sent. The above extra information is encapsulated into a data structure, called *secure key*, and inserted into the head of the ESP tokenset.

Definition 2. *(Secure Key).* Let v_i^S be the node processing the ESP tokenset, v_{i+1}^S be a contact of v_i^S, $(K_{v_i^S} \cdot B)$ be the public key of v_i^S, $(K_{v_{i+1}^S} \cdot B)$ be the public key of the neighbor v_{i+1}^S, $sessionID_{(v_i^S, v_{i+1}^S)}$ be the session identity of v_i^S and v_{i+1}^S. *SecureKey* between v_i^S and v_{i+1}^S is defined as follows:

$$SecureKey_{(v_i^S, v_{i+1}^S)} = Enc_{(K_{v_{i+1}^S} \cdot B)}(ID(v_i)||(K_{tempv_{payer}^S} \cdot B) ||sessionID_{(v_i^S, v_{i+1}^S)})$$

Example 5. Let us continue with Example 3. After receiving the request and Validator from Bob, Evans generates a pair of temporary public and private keys, $(K_{tempEvans} \cdot B, K_{tempEvans})$, to be used for the current payment transaction only. Evans keeps the temporary private key, and propagates the temporary public key to his contacts, that is, Alice and Walker. In particular, Evans encrypts his temporary public key, the session IDs between him and them (i.e., $sessionID_{(Evans,Alice)}$, $sessionID_{(Evans,Walker)}$), and his ID (i.e., $ID(Evans)$) with their public keys, i.e., $(K_{Alice} \cdot B)$ and $(K_{Walker} \cdot B)$ respectively, and obtains two distinguished encryptions. From the above information, Evans creates two secure keys, as follows:

- $SecureKey_{(Evans,Alice)} = Enc_{(K_{Alice} \cdot B)}(ID(Evans)||K_{tempEvans} \cdot B) ||sessionID_{(Evans,Alice)})$
- $SecureKey_{(Evans,Walker)} = Enc_{(K_{Walker} \cdot B)}(ID(Evans)||K_{tempEvans} \cdot B) ||sessionID_{(Evans,Walker)})$

The secure key (Definition 2), the SmartPay tokenset as defined in [22, 23], and the Validator (Definition 1) are then enveloped into a unique data structure, namely, the *ESP*

tokenset. Moreover, to guarantee the integrity of the content of elements of ESP tokenset, i.e., no intermediate nodes can modify the public key encryption, a signature of ESP tokenset encryption is made with the private key of the sending contact using the Elliptic Curve Digital Signature Algorithm (ECDSA) algorithm [69]. An ESP tokenset is formally defined as follows:

Definition 3. *(ESP Tokenset).* Let v_i^S be a node, v_{i+1}^S be one of v_i^S's contacts, v_{payer}^S be the payer, v_{payee}^M be the payee. Validator is created by v_{payee}^M as in Definition 1. SecureKey is created by v_i^S as in Definition 2, SPTokenset is the tokenset as defined in [22, 23]. Hence, the ESP tokenset is defined as follows:

$$TK_{(v_{payer}^S, v_{i+1}^S)} = [combi||sign_{K_{v_i^S}}(combi)]$$

where

$$combi = [SecureKey_{(v_i^S, v_{i+1}^S)}||(SPTokenSet_{(v_{payer}^S, v_{i+1}^S)}) \\ ||Validator_{v_{payee}^M}]$$

and $sign()$ is the function that v_i^S uses for generating the signature of the encryption of ESP tokenset with its private key.

Example 6. Let us continue with Example 5. After creating the secure key, Evans detaches Validator from the received request from Bob, and creates two tokensets for his two contacts on SSNL, i.e., Alice and Walker, as follows:

- $TK_{(Evans,Alice)} = combi||sign_{K_{Evans}}(combi)$, where $combi = [SecureKey_{(Evans,Alice)} ||SPTokenset_{(Evans,Alice)}||Validator_{Bob}]$.
- $TK_{(Evans,Walker)} = combi||sign_{K_{Evans}}(combi)$ where $combi = [SecureKey_{(Evans,Walker)} ||SPTokenset_{(Evans,Walker)}||Validator_{Bob}]$.

where $SecureKey_{(Evans,Alice)}$, $SecureKey_{(Evans,Walker)}$ are like in Example 5, whereas Validator is like in Example 3.

v_{payer} sends the initial ESP tokenset to its contacts and waits for the final ESP tokenset from v_{payee}. Once received, v_{payer} decrypts the SmartPay tokenset, and enforces the trust preferences according to the Smartpay protocol.

Intermediate nodes. Before presenting in details the steps executed by an intermediate node in the ESP protocol, let us remind that, when nodes move around, MANET can be grouped into many partitions. In order for them to connect to each other, connectors forward ESP tokensets from a partition to nodes of another partitions in their radio ranges. In a

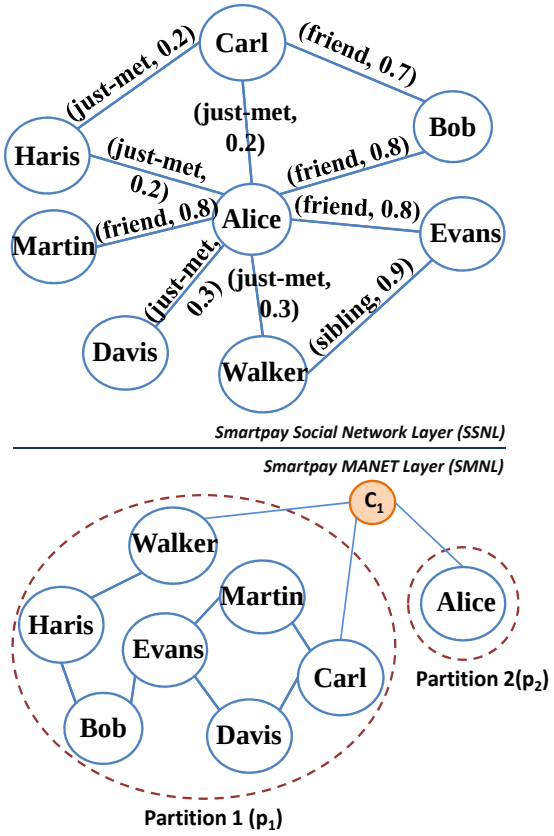


Fig. 4.4 Social Network Layer and Manet Layer Cooperation

MANET partition, nodes that do not have any neighbor to propagate the tokenset are called *border nodes*. These nodes send the received ESP tokenset to connectors located in their radio range. There might be more than one border node in a partition.

Example 7. Let us continue with Example 2 by assuming that the MANET is divided into two partitions, as in Figure 4.4. Suppose that Walker stays in partition p_1 and receives from Haris an ESP tokenset whose destination is Bob. However, Walker does not have any neighbor in p_1, so he forwards the ESP tokenset to connector C_1. In this example, Walker is a border node. When C_1 receives the ESP tokenset from Walker, it forwards the received ESP tokenset to nodes located in the radio range of C_1, that is, nodes in partition p_2. In this example, it is assumed that Alice is a node in p_2 staying in the radio range of C_1, so Alice is a border node as well. She can receive the ESP tokenset from p_1 through C_1. Alice continues to propagate the received ESP tokenset to her contacts. The ESP tokenset is then propagated until it arrives to Bob.

Now let us describe the operation done by intermediate nodes, encoded by Algorithm 1. Let us assume that the intermediate node v_i receives an ESP tokenset, i.e., $TK_{(v_{payer},v_i)}$, from node v_{i-1}. Let $TK_{(v_i,v_{i+1})}$ be the ESP token aggregated with the information of the relationship between v_i and v_{i+1} by v_i. The result of Algorithm 1 is stored into a variable, namely *i_resProc*. This variable can assume one of the values in the set $\{\_SENT\_PAYER, \_NO\_CONTACT, \_SUCCESS\_FORWARD, \_SUCCESS\_AGGR\_PROP, \_DUP\_TKSET, \_CHANGED\_CONTENT\}$, where *_SENT_PAYER* means that the ESP tokenset is sent back to the payer, *_NO_CONTACT* means that there does not exist any contact for v_i to forward the ESP tokenset to, *_SUCCESS_FORWARD* means that v_i forwards the ESP tokenset to its neighbors, *_SUCCESS_AGGR_PROP* means that v_i aggregates the information on the relationship between itself and its contacts into the received ESP tokenset and forwards the aggregate tokenset to its contacts, *_DUP_TKSET* means that the tokenset was processed before, whereas, *_CHANGED_CONTENT* means that the tokenset content has been changed on the communication channel. The values *_SUCCESS_AUTH* and *_FAILED_AUTH* are used in the algorithm to show the result of tokenset authentication. *_SUCC_AUTH* shows that the tokenset is authenticated successfully, whereas *_FAILED_AUTH* means that the authentication fails. We also use *_SUCCESS_DECRYPT* to describe the status where SecureKey decryption is successful.

We assume that each user on SSNL has the public keys of his/her contacts. v_i first decrypts the SecureKey received tokenset with its private key (line 2) so as to determine if it is the intermediate node needing to aggregate the relationship information between itself and v_{i-1}. In case the decryption fails, v_i is not the contact of the sender, v_i needs to transfer the tokenset to its neighbors or connectors on SMNL (line 33). If the decryption works

Function 1 *processTokenset()*

Input: $TK_{(v_{payer},v_i)}$
Output: *i_resProc*

Let *seckey* be decryption of $TK_{(v_{payer},v_i)}.SecureKey$.
$resDec = decryptECC(TK_{(v_{payer},v_i)}.SecureKey, K_{v_i}, seckey)$;
if $(resDec == \_SUCCESS\_DECRYPT)$ **then**
  $K_{v_{i-1}} \cdot B = findContactPbKey(secKey.ID(v_{i-1}))$;
  $resAuthen = authenKey(TK_{(v_{payer},v_i)}, (K_{v_{i-1}} \cdot B))$;
  if $(resAuthen == \_SUCCESS\_AUTH)$ **then**
    if $isDuplicate(seckey.SessionID_{(v_{i-1},v_i)})$ **then**
      if $(isDestination(TK_{(v_{payer},v_i)}.Validator, K_{v_i}))$ **then**
        $send(TK_{(v_{payer},v_i)}, v_{payer})$;
        $i\_resProc = \_SENT\_PAYER$;
      else
        if $(hasContacts())$ **then**
          $TK_{(v_{payer},v_{i+1})} = aggregateTokenset(TK_{(v_{payer},v_i)})$;
          $i\_resProc = sendOnSMNL(TK_{(v_{payer},v_{i+1})})$;
          return i_resProc;
        else
          $Drop(TK_{(v_{payer},v_i)})$;
          $i\_resProc = \_NO\_CONTACT$;
          return i_resProc;
        end if
      end if
    else
      $Drop(TK_{(v_{payer},v_i)})$;
      $i\_resProc = \_DUP\_TKSET$;
      return i_resProc;
    end if
  else
    $Drop(TK_{(v_{payer},v_i)})$;
    $i\_resProc = \_FAILED\_AUTH$;
    return i_resProc;
  end if
end if
$i\_resProc = sendOnSMNL(TK_{(v_{payer},v_i)})$;
return i_resProc;

out (line 3), v_i is the node needing to do aggregation. However, it also needs to verify the integrity of the received tokenset and that the tokenset is truly from its SSNL contact, by using the public key of its SSNL contact (line 5). However, v_i does not know which contact sent the tokenset to it. Therefore to get the public key of its sending contact, v_i retrieves the ID of the sending contact in the decryption. Assuming that, the found ID is of v_{i-1}, i.e., $ID(v_{i-1})$, it then looks for the public key of v_{i-1} (line 4). Then v_i authenticates the tokenset. If the authentication works out, it means that the integrity of the received tokenset is guaranteed and the tokenset content has been not changed. Hence, v_i needs to check if the tokenset is duplicated based on the *SessionID* in SecureKey (line 7). If the authentication or the duplication checking fail, the tokenset is dropped (lines 23, 28). If the duplication checking is successful, the next step is to evaluate the Validator, if v_i is the destination of the tokenset (line 8). In particular, v_i uses its private key to decrypt the Validator and then recovers the shuffled value in the decrypted Validator and sees if it makes sense or not. If the result contains the local information of v_i, it proves that v_i is the destination of the tokenset, it then sends the tokenset back to the payer (line 9). Otherwise, v_i obtains the contact list (line 12) by retrieving from its local storage, if it has contacts it does aggregation between itself and its contacts by calling function *aggregateTokenset()* (line 13) (see Function 2), then calls function *sendOnSMNL()* (line 14) (see Function 3) to propagate the tokenset. If it does not have any contact, the social path through it is dead-end, so it drops the tokenset (line 17).

Function 2 *aggregateTokenset()*

Input: $TK_{(v_{payer},v_i)}$
Output: $TK_{(v_{payer},v_{i+1})}$

1: $TK_{(v_i,v_{i+1})}.SecureKey = encryptECC(concatenate(ID(v_{i+1}), K_{tempv_{payer}} \cdot B, generateSessionID(v_i,v_{i+1})), K_{v_{i+1}} \cdot B)$;
2: $TK_{(v_i,v_{i+1})}.SPTokenset = SmartPay.createTokenset(TK_{v_{payer},v_i}.SPTokenset, K_{tempv_{payer}} \cdot B)$;
3: $TK_{(v_i,v_{i+1})}.Validator = TK_{(v_{payer},v_i)}.Validator$;
4: $TK_{(v_{payer},v_{i+1})} = concatenate(TK_{(v_i,v_{i+1})}, signECDSA(TK_{(v_i,v_{i+1})}, K_{v_i}))$;
5: $return\ TK_{(v_{payer},v_{i+1})}$;

Additionally, function *aggregateTokenset()* receives an input, that is, a tokenset from v_{i-1}. It then returns an output, that is, a tokenset $TK_{(v_{payer},v_{i+1})}$ aggregated the relationship information on the edge connecting it and its contact, i.e., v_{i+1}. v_i generates the session ID between it and v_{i+1} then concatenates the ID of v_{i+1}, the temporary public key of the payer, and the session ID. After that, v_i encrypts this concatenation by the public key of v_{i+1} (line 1). Then, v_i aggregates the relationship information between it and v_{i+1}, using the temporary public key of the payer, by calling function *SmartPay.createTokenset()* as in [22, 23] (line 2).

The Validator is reused from the received tokenset without being modified (line 3). After that, v_i creates a signature of the aggregate tokenset and concatenate the new tokenset and its signature (line 4).

Function 3 *sendOnSMNL()*

Input: $TK_{(v_{payer},v_j)}$
Output: *resSend*
1: $resSend = false$;
2: **if** ($hasNeighbor()$) **then**
3: $resSend = send(TK_{(v_{payer},v_j)}, list\_manet\_neighbor)$;
4: **else**
5: $resSend = send(TK_{(v_{payer},v_j)}, list\_manet\_connector)$;
6: **end if**
7: *return resSend*;

Function *sendOnSMNL()* receives a tokenset $TK_{(v_{payer},v_j)}$. v_i verifies if it has any neighbors (line 2) by the neighborhood discovery protocol.[1] In case the verification works out, v_i obtains a list of neighbors (i.e., $list\_manet\_neighbor$), then transfers the tokenset to neighbors in $list\_manet\_neighbor$ (line 3). Otherwise, $hasNeighbor()$ returns a list of connectors (i.e., $list\_manet\_connector$) to v_i, then v_i sends the tokenset to its connectors (line 5).

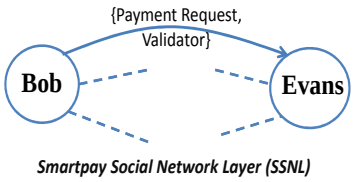


Fig. 4.5 Step 1 of Example 8

The following example clarifies how Algorithm 1 works.

Example 8. Let us continue with Example 2 and Figure 4.4. For the sake of simplicity, we do not consider any trust preference, so the relationship information is removed from the figures depicting the steps in the example. Consider the path on SSNL connecting Bob to Evans, that is, Evans→Alice→Bob, where Evans is the payer, and Bob is the payee. After Bob makes a request to Evans for an amount of money, Evans sends the initial ESP tokenset back to Bob. There is only one SMNL path from Evans to Bob, that is, Evans→Bob. According to Algorithm 1, the following steps are performed:

[1]Mobile Ad Hoc Network (MANET) Neighborhood Discovery Protocol (NHDP) (RFC 6130) https://tools.ietf.org/html/rfc6130.

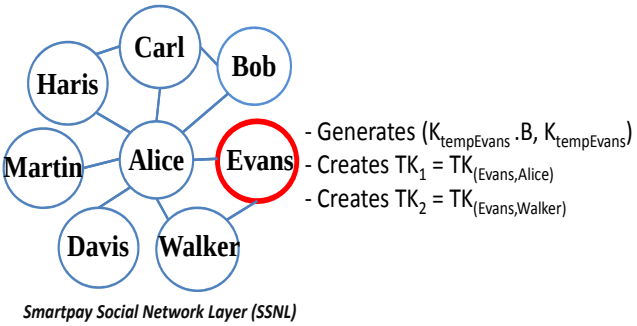


Fig. 4.6 Step 2 of Example 8

1. Bob sends the payment request and Validator to Evans (see Figure 4.5).

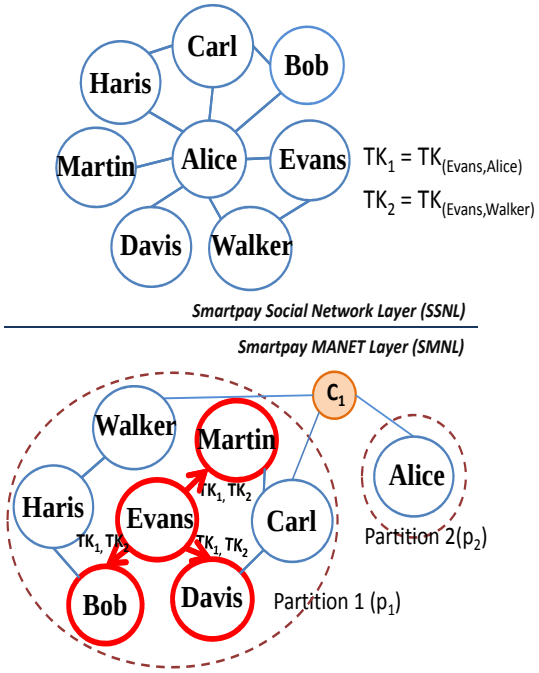


Fig. 4.7 Step 3 of Example 8

2. Evans receives the request from Bob (see Figure 4.6), then it is:

   - Evans generates a pair of temporary keys $(K_{tempEvans} \cdot B, K_{tempEvans})$ only used for this payment transaction.
   - Evans initializes the ESP tokensets for his two directly connected contacts on SSNL, i.e., Alice and Walker, to obtain $TK_1 = TK_{(Evans,Alice)}$ and $TK_2 = TK_{(Evans,Walker)}$.

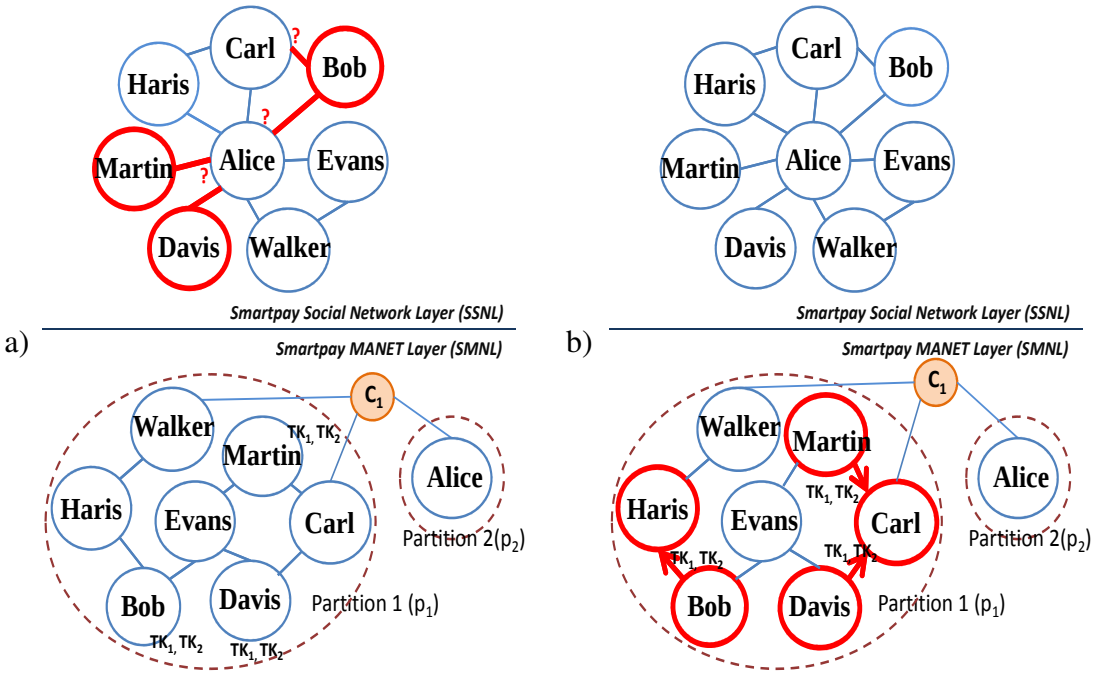


Fig. 4.8 Step 4 of Example 8

3. Evans then looks for the list of his neighbors to forward these two ESP tokensets. Evans has three neighbors, that is, Martin, Bob, and Davis. Hence, he sends these two ESP tokensets to each of them.

4. Martin, Bob, and Davis receive two ESP tokensets. They start to verify if they are contacts of Evans in the social graph (line 5 of Algorithm 1). It results that they are not Evans' contacts (see step 4a in Figure 4.8-a). Bob, Martin, and Davis thus look for their neighbors (line 12), and forward the received ESP tokensets to them without changing the content (line 33). Bob sends two ESP tokensets to Haris while Martin and Davis send two ESP tokensets to Carl (see step 4b in Figure 4.8-b).

5. After receiving the ESP tokensets, Carl and Haris verify if they are contacts of Evans by decrypting the SecureKey (line 2 of Algorithm 1) (see step 5a in Figure 4.9-a). The function fails, therefore, Carl and Haris forward the ESP tokensets to their neighbors (see step 5b in Figure 4.9-b). Haris' neighbor is Walker, thus, Haris transfers the ESP tokensets to Walker. Carl does not have neighbors, so he sends the ESP tokensets to connectors in his radio range. Here we assume that Carl's connector is C_1.

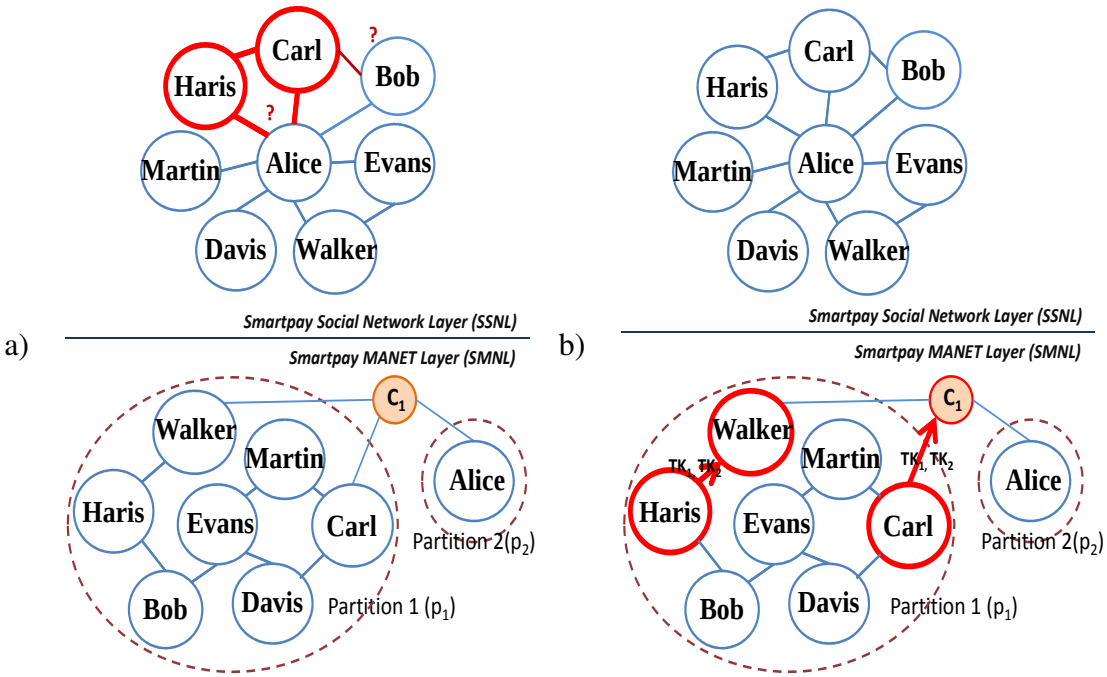


Fig. 4.9 Step 5 of Example 8

6. Walker receives two ESP tokensets from Haris. He checks if he is Evans' contact (line 2 of Algorithm 1) and the decryption succeeds (see step 6a in Figure 4.10-a).
Then he authenticates if the tokenset content has been changed (line 5 of Algorithm 1). Assume that the tokenset has been changed on the communication channel between Walker and Haris. Hence, Walker checks if this tokenset from Evans was processed before (line 7 of Algorithm 1). Assume that the tokenset has been already processed. So, Walker continues to verify if he is the payee of Evans (line 8) and the check fails. Therefore, Walker continues to search his contacts (line 12) and it retrieves Alice. Walker aggregates the relationship information between him and Alice to gain $TK_3 = TK_{(Evans,Alice)}$ (line 13). Then, Walker needs to propagate the tokenset (line 14), he looks for his neighbors, he finds out that he does not have neighbors. So, he forwards to C_1 the ESP tokensets, including the updated ESP tokenset $TK_3 = TK_{(Evans,Alice)}$ and the remaining tokenset TK_1 from Haris (see step 6b in Figure 4.10-b).

7. Two ESP tokensets from Walker and one from Carl are for Alice. Another one is not for Alice, that is, $TK_2 = TK_{(Evans,Walker)}$ from Carl. So, Alice should forward TK_2 to her neighbors without modifying its content, but Alice does not have any neighbor. So, she forwards TK_2 to C_1.

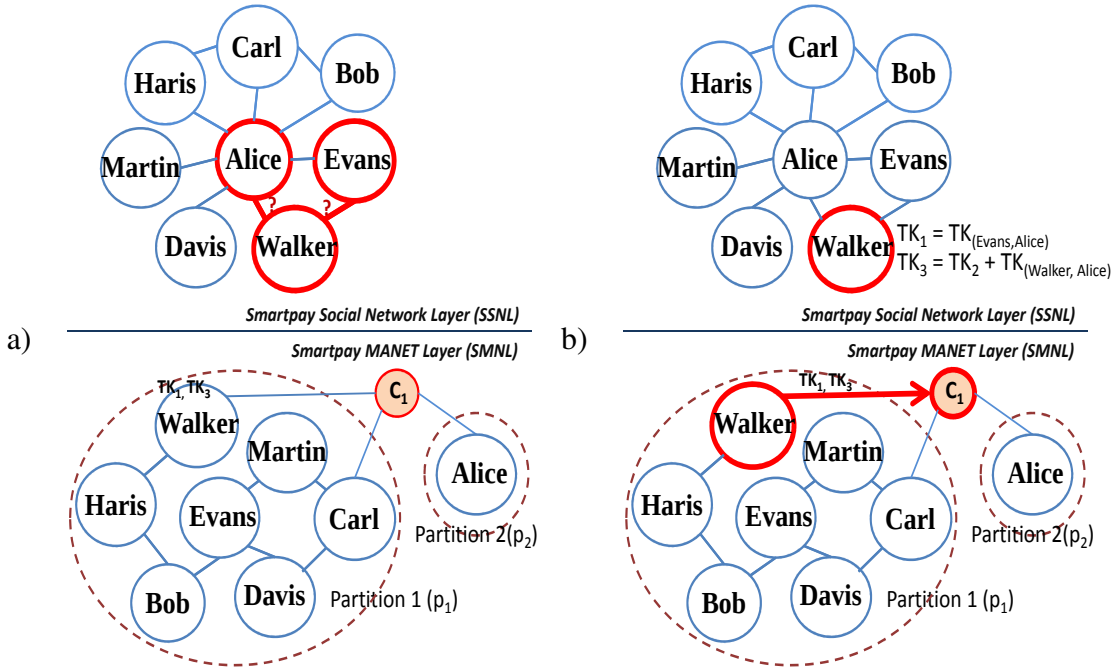


Fig. 4.10 Step 6 of Example 8

8. Connector C_1 sends the received ESP tokensets from Carl and Walker to partition p_2. In p_2, there is only one node, that is, Alice (see step 7 in Figure 4.11).

9. With the remaining tokensets, Alice searches the list of her contacts (line 12) and finds out six contacts that have to update their contents, that is, Bob, Carl, Davis, Haris, Martin, and Evans. She updates the received ESP tokensets (line 13), and obtains $TK_4 = TK_{(Evans,Bob)} = TK_1 + TK_{(Alice,Bob)}$, $TK_5 = TK_{(Evans,Carl)} = TK_1 + TK_{(Alice,Carl)}$, $TK_6 = TK_{(Evans,Davis)} = TK_1 + TK_{(Alice,Davis)}$, $TK_7 = TK_{(Evans,Haris)} = TK_1 + TK_{(Alice,Haris)}$, $TK_8 = TK_{(Evans,Martin)} = TK_1 + TK_{(Alice,Martin)}$, $TK_9 = TK_{(Evans,Walker)} = TK_1 + TK_{(Alice,Walker)}$, $TK_{10} = TK_{(Evans,Bob)} = TK_3 + TK_{(Alice,Bob)}$, $TK_{11} = TK_{(Evans,Carl)} = TK_3 + TK_{(Alice,Carl)}$, $TK_{12} = TK_{(Evans,Davis)} = TK_3 + TK_{(Alice,Davis)}$, $TK_{13} = TK_{(Evans,Haris)} = TK_3 + TK_{(Alice,Haris)}$, $TK_{14} = TK_{(Evans,Martin)} = TK_3 + TK_{(Alice,Martin)}$, $TK_{15} = TK_{(Evans,Walker)} = TK_3 + TK_{(Alice,Walker)}$. Alice wants to send them to her neighbors (line 14), so searches for them, but she does not have any neighbor. Therefore, she forwards all the tokensets to connector C_1 which is in her radio range (see step 9 in Figure 4.13).

10. C_1 receives the ESP tokensets from Alice, and sends all of them to Walker and Carl who are border nodes of partition p_1. Let us first consider Walker. Carl will repeat similar steps to process the received ESP tokensets (see step 10 in Figure 4.14).

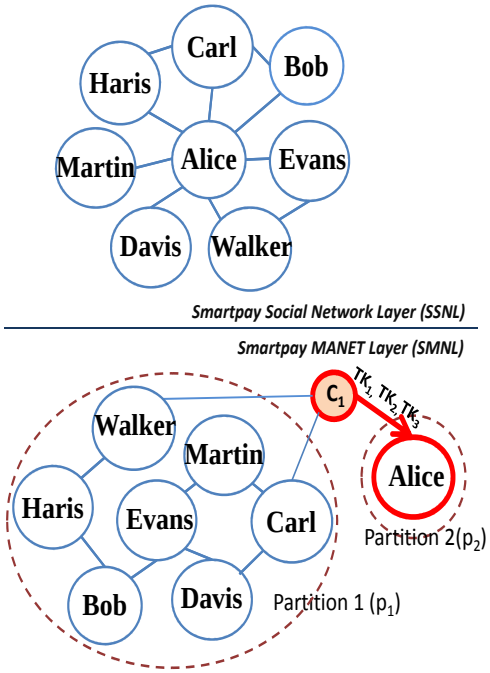


Fig. 4.11 Step 7 of Example 8

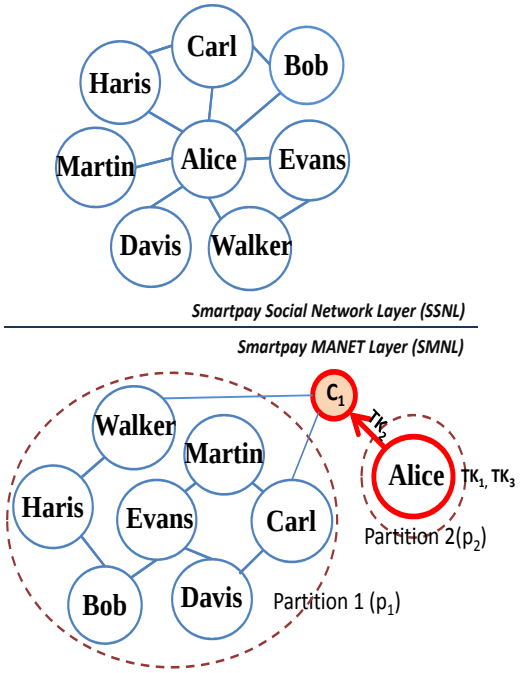


Fig. 4.12 Step 8 of Example 8

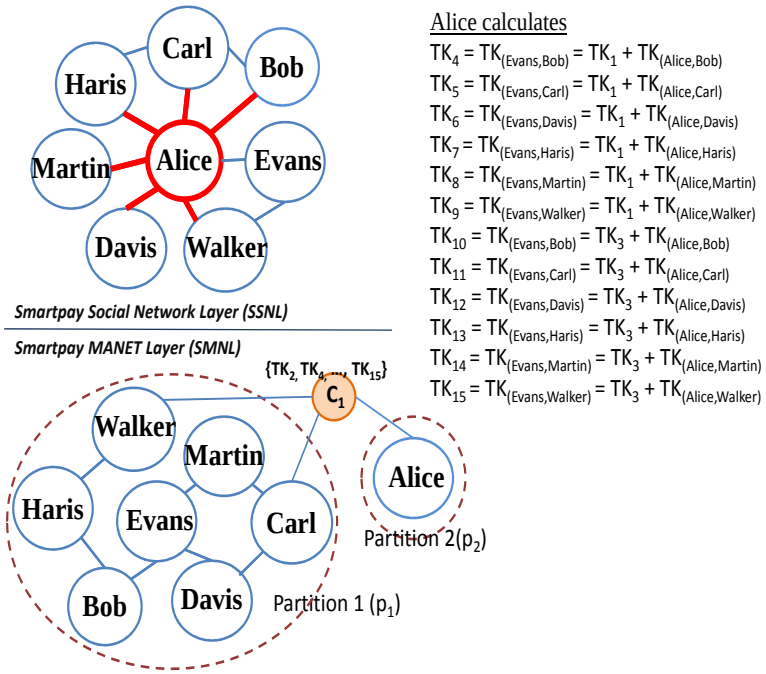


Fig. 4.13 Step 9 of Example 8

11. Walker receives the ESP tokensets from C_1. He decrypts the SecureKey of all received tokensets to check if there is any tokenset for him to update (lines 2, 5 of Algorithm 1). But no tokenset is for him. Then, he forwards all of them to Haris who is his neighbor (line 33) (see step 11a in Figure 4.15-a). Among the received ESP tokensets from Walker, there are tokensets for Haris to update, that is, TK_7 and TK_{13}. Haris forwards the other tokensets to Bob as Bob is his neighbor (see step 11b in Figure 4.15-b).

12. Bob decrypts SecureKey of the received tokensets (line 2 of Algorithm 1), there is one tokenset for him. He authenticates if the tokenset has been changed on the communication channel (line 5 of Algorithm 1), then he also checks if the tokenset has been processed (line 7 of Algorithm 1). Then, he validates if he is the destination of the received ESP tokensets (line 8). He is the destination of two ESP tokensets TK_4 and TK_{10}. (see step 12 in Figure 4.16).

13. Bob sends ESP tokensets to Evans (line 9) (see step 13 in Figure 4.17).

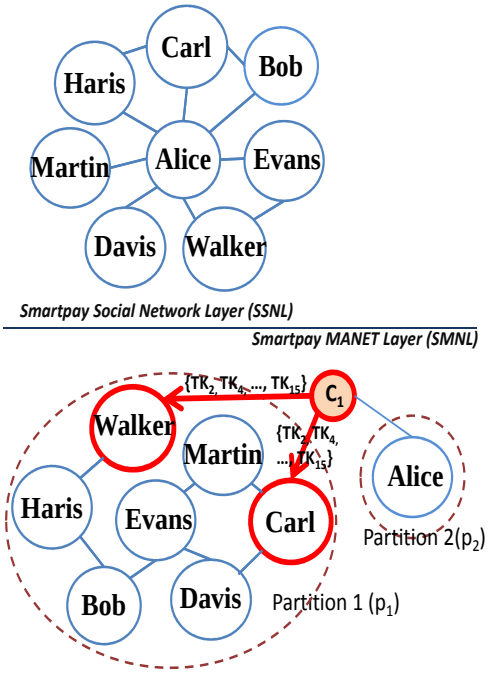


Fig. 4.14 Step 10 of Example 8

4.4 Condition-driven flooding

As presented in Chapter 3, SmartPay exploits k-anonymity [137] to reduce the number of distributed tokensets. Relationship information privacy as well as the number of propagated tokensets depend on the value of k. Selecting a reasonable value of k to make the network consumption effective is not easy. In particular, in case k is large, network performance is similar to the one of broadcasting techniques, but privacy is preserved more strongly. In case k is small, the bandwidth consumption is low but privacy cannot be preserved effectively. Therefore, in order to preserve privacy, and, at the same time, improve network performance, here we propose an alternative approach exploiting secure comparison techniques.

Let us first recall that the goal of finding a social path connecting the payee and the payer is to verify if the relationship between the payee and the payer satisfies the specified trust preferences. Let us consider a trust condition $tc = (rt, d_{max}, t_{min})$ in a trust preference of the payer. Finding a valid social path connecting the payer and the payee requires to evaluate whether the relationship type of the path is equal to rt, the aggregated trust value is greater than or equal to t_{min}, and the depth of the path is less than or equal to d_{max}. As such, if an intermediate node is able to detect that one of these conditions is not satisfied, it is able to terminate the ESP tokenset propagation. Thus, the number of propagated ESP tokensets is

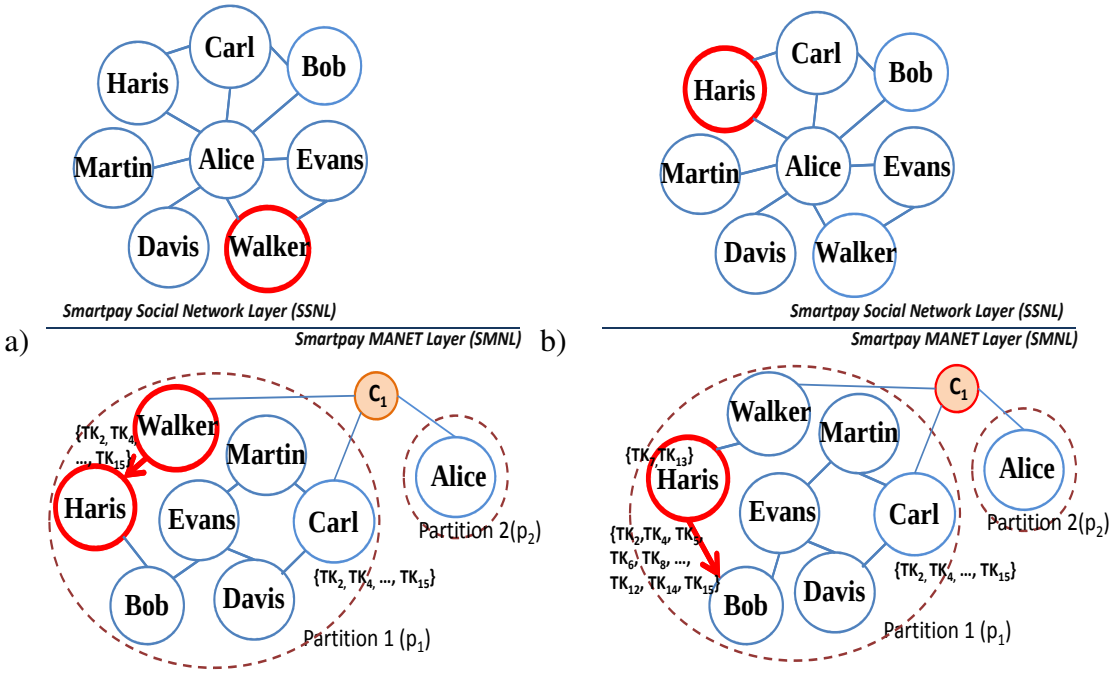


Fig. 4.15 Step 11 of Example 8

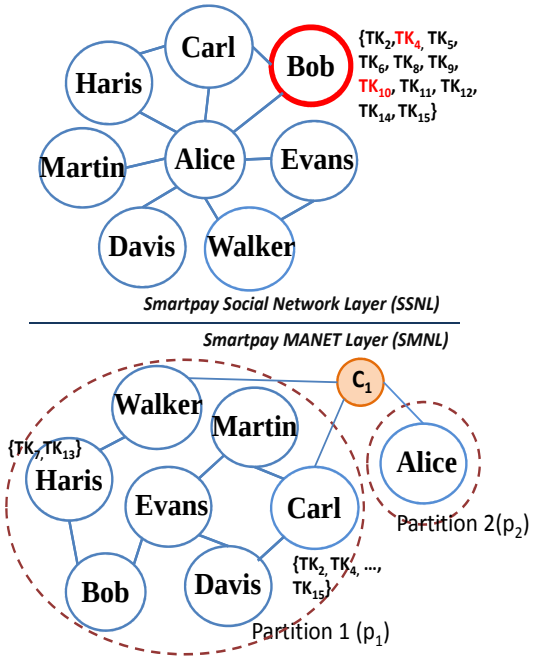


Fig. 4.16 Step 12 of Example 8

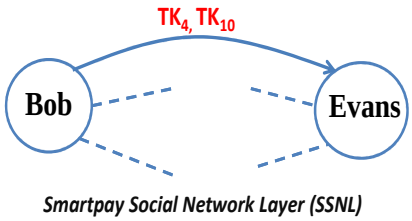


Fig. 4.17 Step 13 of Example 8

significantly reduced. For this purpose, there is the need of a way that enables intermediate nodes to privately verify trust conditions. In [22, 23], since the trust condition verification is enforced at the payer side, only the payer can read trust conditions. In this chapter, we assume that each intermediate node receives trust conditions encrypted with the payer's temporary public key, so they cannot learn plain texts of trust conditions. With respect to the protocol explained in Section 4.3, we add an additional step. The intermediate node, say v_i, has to evaluate the updated ESP tokensets based on the encrypted trust conditions by exploiting the proposed secure comparison approach. Based on the results, v_i, propagates only the updated ESP tokensets that verify the trust conditions.

Example 9. Let us continue with Example 2. Assume that an intermediate node v_i receives an ESP tokenset along with the encryption of trust condition $tc = (Friend, 3, 0.6)$. After receiving the request from Bob, Evans initializes two ESP tokensets for his contacts, Alice and Walker, where the trust values are 0.8 and 0.9, respectively. These trust values and the depths (i.e., 1, as they are direct friends) satisfy tc. However, the type of the relationship in the initial ESP tokenset is *Friend* between Alice and Evans, but is *Sibling* between Evans and Walker. Hence, Evans only sends the ESP tokenset to Alice. The number of forwarded ESP tokensets is reduced by half. In turn, Alice has six contacts, that is, Bob, Carl, Haris, Martin, Davis, and Walker. She creates six duplicates of the received ESP tokenset, then updates them with the relationship information of the edges between herself and these contacts. After doing the secure comparison between the encrypted trust conditions and each of the updated ESP tokensets, it results that, among these six contacts, Alice chooses Bob and Martin to propagate the updated ESP tokensets since the relationship information in the ESP tokensets for both Bob and Martin are $(Friend, 2, 0.64)$. Here, the number of forwarded ESP tokensets is also reduced. In the four other updated ESP tokensets, the type of the relationship is not *Friend*. Therefore, Alice sends the updated ESP tokensets only to Bob and Martin.

In order to do a comparison between the encrypted trust conditions and the encrypted values in the ESP tokenset, we adopt the secure comparison scheme proposed by F. Kerschbaum

et al. in [74]. The authors proposed a general framework for a secure comparison between x_0 and x_1. All participants involved in this scheme do not know the real values of x_0 and x_1. They just receive the encrypted values, denoted as $E(x_0)$ and $E(x_1)$. This framework exploits homomorphic encryption, and the authors used *RSA* (1024-bit key length) for the experiment. However, *RSA* consumes a lot of memory, CPU and time on encrypting and decrypting, so it is not suitable for the mobile environment. To cope with this issue, we adopt the original scheme in [74], and apply the homomorphic binary ECC algorithm into it.

Secure comparison is done on trust, depth, and type of the relationship. Among these three components, trust and depth are numbers, whereas, the type needs to be converted into a number in the range $[1,35]$. This range is chosen according to the quantity of relationship types defined in the widely used FOAF vocabulary [32]. The basic idea is that, for each intermediate node, instead of forwarding a received ESP tokenset to several contacts, this node applies the secure comparison scheme in [74] to compare the encrypted trust, depth, type of the relationship in the received tokenset with the respective encrypted threshold in the trust condition. Based on the result, the node can decide to which contacts the ESP tokenset should be forwarded.

Let us denote trust, depth, type in a trust condition as t_{min}, d_{max}, rt, respectively. These are encrypted with the payer's temporary public key (i.e., $(k_{v_{payer}} \cdot B)$), and denoted respectively as $E_{v_{payer}}(t_{min})$, $E_{v_{payer}}(d_{max})$, and $E_{v_{payer}}(rt)$. The encryptions are sent from the payer to its contacts, say v_i. Let $T_{(v_{payer},v_i)}$ be the ESP tokenset containing the encryption of trust, depth, and type which are collected in the social path from the payer to v_i. Let $T_{(v_{payer},v_i)}.trust$, $T_{(v_{payer},v_i)}.depth$, and $T_{(v_{payer},v_i)}.rt$, $E_{v_{payer}}(T_{(v_{payer},v_i)}.trust)$, $E_{v_{payer}}(T_{(v_{payer},v_i)}.depth)$, $E_{v_{payer}}(T_{(v_{payer},v_i)}.rt)$ be respectively trust, depth and type tokenset in $T_{(v_{payer},v_i)}$ and their corresponding encryptions with the payer's public key. Let us first consider secure comparison on trust. Secure comparison on depth and relationship type are managed in a similar way. Let us assume that v_i sends $E_{v_{payer}}(T_{(v_{payer},v_i)}.trust)$ to v_{i+1}. Node v_{i+1} aggregates $T_{(v_i,v_{i+1})}.trust$ (i.e., the trust value of the edge between v_i and v_{i+1} into $E_{v_{payer}}(T_{(v_{payer},v_i)}.trust)$ and obtains $E_{v_{payer}}(T_{(v_{payer},v_{i+1})}.trust)$. Then, it applies the scheme in [74] to decide if this updated ESP tokenset can be propagated to its contacts by comparing two encryptions of the trust threshold and the aggregate trust in the tokenset. To do this comparison, v_{i+1} randomizes two large numbers r, r' in $\mathbb{N}$, and calculates the value of $E_{v_{payer}}(c)$ with the following equation, where $E(c)$ is an encryption for the computation at the next round as described in [74].

$$
\begin{aligned}
E_{v_{payer}}(c) &= r \cdot (E_{v_{payer}}(t_{min}) - E_{v_{payer}}(T_{(v_{payer},v_j)}.trust)) \\
&\quad - E_{v_{payer}}(r') \\
&= (r \cdot r_0 \cdot B - r \cdot r_1 \cdot B - r'' \cdot B, (r \cdot (t_{min} \\
&\quad - T_{(v_{payer},v_j)}.trust) - r') + (r \cdot r_0 - r \cdot r_1 - r'') \\
&\quad \cdot k_{v_{payer}} \cdot B) \\
&= E_{v_{payer}}(r \cdot (t_{min} - T_{(v_{payer},v_j)}.trust) - r')
\end{aligned}
\tag{4.1}
$$

where r_0, r_1, r'' are the randoms generated while the encryptions are done.

The basic idea of Formula 4.1 is to compute $d = (t_{min} - T_{(v_{payer},v_{i+1})}.trust)$. d is used for comparing t_{min} and $T_{(v_{payer},v_{i+1})}.trust$. If $d < 0$, it implies that $t_{min} < T_{(v_{payer},v_{i+1})}.trust$; otherwise, $t_{min} \geq T_{(v_{payer},v_{i+1})}.trust$. However, to hide d, d is multiplied with a random number r to obtain a multiplication $m = d.r$. Then, to prevent the factoring of the result [76], r' is added to m and we obtain $c = d.r + r'$. Here, we can replace the addition with the subtraction, and we have $c = d.r - r'$. Actually r, r' are used for obfuscating the value of d. However, the goal is to do a secure comparison between t_{min} and $T_{(v_{payer},v_{i+1})}.trust$. Therefore, c must be encrypted, and we obtain $E(c)$ as above.

After computing $E_{v_{payer}}(c)$, v_{i+1} sends $(a_1, a_2, a_3) = (E_{v_{i+1}}(0), E_{v_{i+1}}(1), E_{v_{payer}}(c))$ back to v_i, where $E_{v_{i+1}}(0)$, $E_{v_{i+1}}(1)$ are encryptions of values '0' and '1' with v_{i+1}'s public key. Then, v_i also randomizes two large numbers r_i, r_i', and flips a coin $b \in \{0,1\}$, then re-calculates (a_1, a_2, a_3):

$$
\begin{aligned}
a_1 &= a_{1+b} + E_{v_i}(0); \\
a_2 &= a_{2-b} + E_{v_i}(0); \\
a_3 &= (-1)^{b \cdot r_i}(a_3) + (-1)^{(1-b)} \cdot r_i' \cdot E_{v_i}(1);
\end{aligned}
\tag{4.2}
$$

b is known only to v_i. v_i sends (a_1, a_2, a_3) back to v_{i+1}. Then, v_i and v_{i+1} collaboratively checks a_3. At v_i side, if $a_3 < 0$, we have the boolean expression $[t_{min} \leq T_{(v_{payer},v_{i+1})}.trust] = 1 - b$; otherwise, we have the boolean expression $[t_{min} \leq T_{(v_{payer},v_{i+1})}.trust] = b$. With the value of b, v_i can learn the result of the comparison. Notice that the boolean expression is 0 when it is false, and it is 1 when it is true.

This secure comparison makes it possible to reduce a large number of messages going through the network, as shown in the performance results reported in Section 4.5.2.

4.5 Experiments

This section presents the performance of ESP protocol and the flooding optimization.

4.5.1 ESP Performance

In order to prove the efficiency of ESP, we measure the time an ESP tokenset spends reaching the payee from the payer in several network configurations. Network configurations are defined based on three parameters: the SMNL and SSNL topologies, the wireless network bandwidth, and the adopted encryption algorithms (i.e., their key sizes).

Regarding the first parameter, we set up several SMNL and SSNL topologies, by varying the number of contacts/nodes from 4 to 17. These results in 16 different network configurations, which are summarized in Table 4.1, where network 1 has the shortest path, including 3 SSNL contacts, and a 8 SMNL hop distance between every 2 SSNL contacts; whereas, network 16 has the longest relationship path, including 9 SSNL contacts, and a 14 SMNL node distance between every 2 SSNL contacts. With respect to the second parameter, we simulate different wireless network bandwidths. In general, MANET nodes transmit data through different wireless standards. We deploy experiments assuming nodes exploit the popular wireless standard IEEE 802.11a/b/g, and their respective bit rates of 54/11/24Mbps. The third parameter that might impact the performance is the payload size of ESP tokenset. Since we are using encryption schemes, the payload might vary based on the key sizes of the adopted algorithms. In particular, we deploy the UDP protocol for exchanging the tokensets between two mobile devices, because its speed and small payload fit well MANET. The UDP payload size varies according to the key sizes of ECC and ECDSA algorithms. In this experiment, we select four ECC key sizes, that is, 163, 283, 409, 571 bits, and two ECDSA key sizes, that is, 384, 521 bits.

Table 4.2 shows the UDP payload size by using different combination of ECC and ECDSA key sizes. The smallest UDP payload size is 446 bytes, using 163 bit ECC key size and 384 bit ECDSA key size, whereas, the greatest UDP payload size is 1298 bytes, using 571 bit ECC key size and 521 bit ECDSA key size. These payload sizes do not exceed the limit of a UDP packet size in MANET, i.e., 1460 bytes. Hence, the ESP tokensets are not segmented into more than one packet.

We used OmneT++[2] and INET framework[3] to set up the different network configurations. OMNeT++ is a component-based C++ simulation library and framework for building network simulators for a variety of application domains, such as sensor networks, wireless ad-

[2]http://www.omnetpp.org/
[3]http://inet.omnetpp.org/

hoc networks, photonic networks, etc. Whereas, the INET Framework is an open-source communication network simulation package for the OMNeT++ simulation environment. The INET Framework contains models for several Internet protocols, such as UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP, IEEE 802.11, MPLS, OSPF. By using these tools, we conduct experiments on physical resources including Duo Core CPU 4GHz, 4GB RAM, 64-bit Windows 7.

Table 4.1 Network configurations

Network	**Number of SSNL contacts**	**Number of SMNL nodes between two SSNL contacts**	**Total number of SMNL nodes**
N1	3	8	17
N2	5	8	33
N3	7	8	49
N4	9	8	65
N5	3	10	21
N6	5	10	41
N7	7	10	61
N8	9	10	81
N9	3	12	25
N10	5	12	49
N11	7	12	73
N12	9	12	97
N13	3	14	29
N14	5	14	57
N15	7	14	85
N16	9	14	113

We measure the delay (ms) spent on transmitting an ESP tokenset from the payer to the payee according to different parameters values. Particularly, the transmission delay includes the time of aggregating an ESP tokenset at each SSNL contact and the time of propagating ESP between SMNL nodes. The experimental results are reported in Figure 4.18. According to Figure 4.18, we can see that at the same bit rate the transmission delays in cases of 384 bit ECDSA are higher than the ones of 521 bit ECDSA, and the transmission delays with increasing ECC key sizes are rising. In the worst case (i.e., 11Mbps bitrate, 521 bit ECDSA and 571 bit ECC), the transmission delay for an ESP tokenset to move through 113 SMNL nodes and to aggregate relationship information among 8 SSNL contacts is 1710,4 ms (approximately 1.7s). This transmission delay is reasonable with MANET

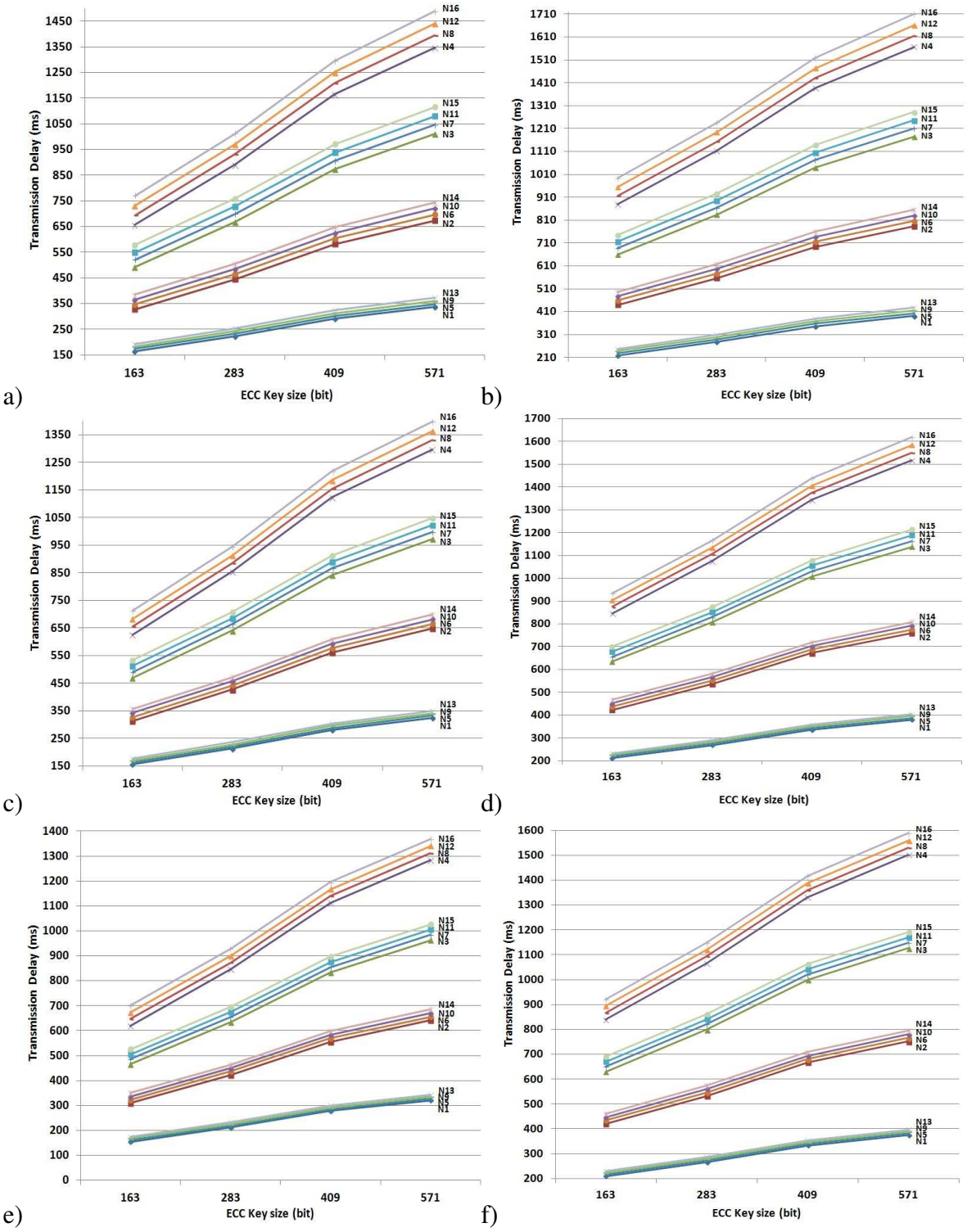


Fig. 4.18 Transmission delay of an ESP tokenset vs key size of ECC and ECDSA vs bit rate standard: a) 11Mbps bitrate and 384bit ECDSA; b) 11Mbps bit rate and 521bit ECDSA; c) 24Mbps bitrate and 384bit ECDSA; d) 24Mbps bit rate and 521bit ECDSA; e) 54Mbps bitrate and 384bit ECDSA; f) 54Mbps bit rate and 521bit ECDSA.

Table 4.2 UDP payload size with different ECC and ECDSA key sizes

ECC key size (Bit)	ECDSA key size (Bit)	UDP payload size (Byte)
163	384	446
283	384	686
409	384	942
571	384	1263
163	521	482
283	521	722
409	521	978
571	521	1298

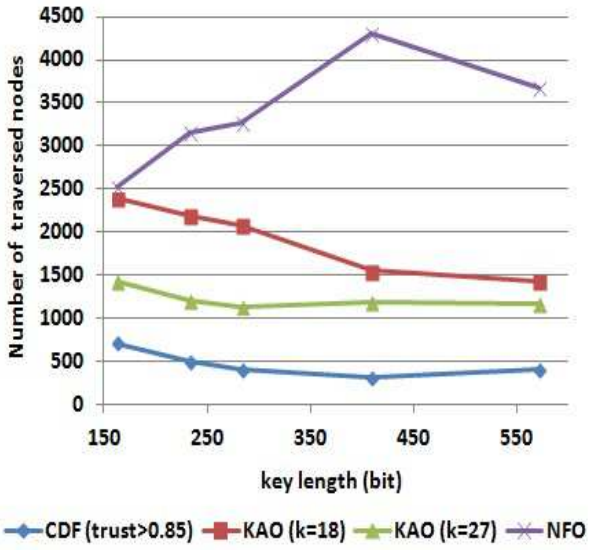
performance requirement. Therefore, this result proves that the proposed ESP has an effective performance.

4.5.2 Condition-driven Flooding Optimization

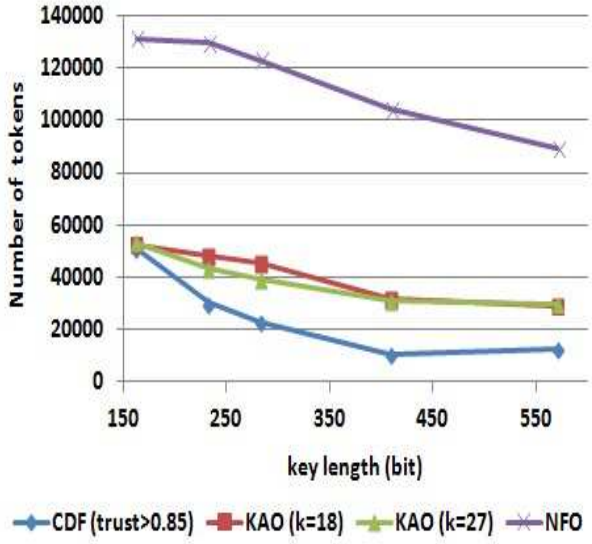
In order to prove the efficiency of the proposed flooding optimization method, we measure the number of generated tokensets propagated through the network and the number of nodes traversed by tokensets, by exploiting the solution proposed in this chapter, i.e., the *C*onditional *D*riven *F*looding (CDF), the solution proposed in [22, 23], i.e., the *K*-*A*nonymity based *O*ptimization (KAO), and the one without any optimization, i.e., the *N*o *F*looding *O*ptimization (NFO). The experiment is performed on the trust value. If the trust threshold is very small, CDF is similar to the broadcast technique. If the trust threshold is very high, the tokensets are easily dropped at each intermediate node. Hence, we choose 85% as a threshold in CDF. This threshold is neither very high nor very small. With KAO, to set a good trade-off we have selected k = 18 and k = 27, that is, half and 3/4 of available relationship types in [32]. In this experiment, we use the Epinion dataset[4]. This is a who-trust-whom online social network of a general consumer review for the site www.Epinions.com. The dataset includes 75.879 nodes and 508.837 edges. We modified the relationship information including a relationship type and a trust value on each edge. Relationship types are uniformly and randomly picked up from [32] based on the ontological FOAF vocabulary specification which provides a set of 35 relationship types. A trust value on an edge e is also randomly generated uniformly from a range [0, 1].

Figure 4.19 shows that CDF flooding has the best performance. In the case of the largest ECC key size (i.e., 512 bits), the number of tokensets flooding the network with KAO

[4]https://snap.stanford.edu/data/soc-Epinions1.html



a) Number of traversed nodes per second



b) Number of flooding tokens per second

Fig. 4.19 A comparison between NFO, KAO, and CDF.

($k \in \{18, 27\}$) is approximately 2.25 times higher than with CDF, whereas the one with NFO is approximately 7 times higher than with CDF; the number of traversed nodes with KAO (k = 18) is approximately 3.48 times higher than with CDF, and the number of traversed nodes with KAO (k = 27) is approximately 2.85 times higher than with CDF; whereas, the one with NFO is approximately 8.9 times higher than with CDF.

4.6 Complexity Analysis

In order to appraise the system efficiency, we analysis the time complexity in two cases, that is, the worst and the best cases. Since this work is a deployment of SmartPay over MANET, which results in the time complexity calculated similarly to in Section 3.7, Chapter 3.

4.7 Security Analysis

Since each ESP tokenset component is encrypted, the tokenset can be considered robust to external eavesdropping. Indeed, the adversaries able to thwart the system are more likely to be playing the roles of participants of the protocol, that is, payer, payee, and intermediate nodes. Based on their behaviors, we classify adversaries into two types, that is, honest-but-curious, and malicious. Honest-but-curious adversaries are intermediate nodes correctly enforcing the protocol by at the same time also looking for extra information. In contrast, malicious nodes might try to modify the ESP token so as to illegitimately retrieve information (e.g., identity, or relationship information). In this section, we discuss how our proposal can resist to both attacks.

4.7.1 Honest-but-curious nodes

According to the honest-but-curious model, it is expected that nodes comply with the proposed protocol and algorithms, by trying to infer additional private information. In this section, we show how ESP is enough robust to avoid the inference of payee's and payer's identities, and relationship information.

Scenario 1: Inferring payee's identity. Payee's identity is stored into the Validator component, which is encrypted with the payee's public key. This makes hard for an adversary to learn its plain-text value. However, an intermediate node might try to infer the payee's identity even without decrypting the Validator component. Indeed, since the node has the list of its neighbors' identities (i.e., IP addresses, MAC addresses), and the respective public keys,

it might try to encrypt neighbor's identity with the corresponding public key and compare the result with Validator content. However, we have to note that this attack is hard to be carried out since Validator's encryption is created based on a large random number generated and held by the payee (see Section 3.1.2), and the timestamp, generated based on the payee's local system and appended to the pre-encrypted Validator. Moreover, to improve the robustness of the system, we adopt the shuffle algorithm in [41] algorithm so as to obtain a permutation of validator content, computed based on a set of random numbers generated and held by the payee. The adversaries cannot have the above mentioned local parameters, hence, they cannot infer the information of the payee's.

Scenario 2: Inferring payer's identity. We recall that in SmartPay [22, 23], in order to elaborate the received tokensets, intermediate nodes have to know the public key of the payer, as such they are aware of the payer identity. To avoid this exposure, in ESP, each payer generates a different temporary public key for every payment transaction instead of using its real public key. Thus, any intermediate node is not able to re-identify the payer by the used public key. According to the proposed protocols, the receiving SMNL node can infer information about the sender (e.g., MAC address, IP address). However, since it cannot see any other information as all other components in the ESP tokenset are encrypted, the receiving SMNL node cannot understand if the sender is indeed the payer and thus the payer indetity.

Scenario 3: Inferring relationship information. Since SPTokenset content is encrypted by the temporary public key of the payer, an intermediate node cannot access the relationship information, as it does not hold the corresponding private key. However, since the node owns the temporary public key of payers, he might try to encrypt a pre-defined set of values for trust and relationship types and then compare the results with encrypted values in SPTokenset. However, let us recall that each node in ESP, including the payer, needs to locally generate one random number for aggregating a trust/type into SPTokenset using the homomorphic ECC encryption (see Section 3.1.2). Random numbers are different at every node, which makes encryptions of the same value different. Hence, even if a node uses the temporary public key of the payer for a statistical analysis, it cannot infer trust/type of SPTokenset as well.

4.7.2 Malicious nodes

Under this attack model, malicious nodes might try to modify the ESP tokenset so as to illegitimately retrieve information (e.g., identity, relationship information). In what follows,

we analyze the scenarios of these malicious attacks and discuss how the proposed protocol resists on them.

Scenario 4: Anti-replay attack. Let us consider the SSNL layer and assume that v_1^S sends a tokenset to v_2^S and that this tokenset passes through node m in the SMNL layer. Let us assume that malicious node m tries to impersonate v_1^S by reusing a copy of the tokenset that has previously passed through m. In SecureKey, a parameter, called sessionID, is inserted and used for determining the processed tokensets. When v_2^S receives the same tokenset, it decrypts SecurKey's encryption with its private key and checks sessionID. In this case, v_2^S verifies that this tokenset has been already processed, then it drops the tokenset, avoiding thus the replay attack.

Scenario 5: Altering tokenset content. According to this scenario, we assume that a malicious node wishes to modify the tokenset content, by replacing one of the elements of the tokenset (i.e., SecureKey, SPTokenset, Validator), and/or by simply inserting/deleting some bits, so as to invalidate the tokenset. To detect the corrupted tokensets, we impose that the ESP is digitally signed by the node processing the ESP tokenset. Since a malicious node does not hold the private key of ESP signature, it cannot generate a new one for the modified ESP tokenset. Thus, the node receiving this ESP tokenset can detect that its content has been maliciously modified, by validating the original digital signature.

Chapter 5

Secure Orchestrated Web Service Composition against Untrusted Broker

Large computing infrastructures, like the Internet, increase the capacity to share information and services across organizations. For this purpose, web services have gained popularity in both research and commercial sectors. A web service is a software system designed to support interoperable application-to-application interactions over the Internet. One of the major goals of web services is to make easier their composition to form more complex services. Two are the main techniques for web service composition, namely, orchestration [33] [106], [65], [75] and choreography [5, 149, 73]. Choreography is a decentralized model with a dynamic sequence of web services used for B2B (Business-to-Business) applications, but it is hard to be specified and implemented. In contrast, orchestration has a BPEL (Business Process Execution Language) engine [110] as a central mediator to control a static sequence of activities described in a BPEL-based file. Such an approach is simpler and easier to be implemented, making orchestration a valid solutions for composite web service deployment. Therefore, in this chapter we will focus on this approach. As such, we assume the workflow underlying the business process is encoded into a BPEL document and processed by a server hosting a BPEL engine (hereafter, the *broker*). According to the orchestration paradigm, the broker coordinates the invocation of services involved in the composition, i.e., partner services, by passing the needed parameters. In general, all previous proposals for the service orchestration model consider the broker as a trusted entity. As such, they never payed attention to the fact that the broker is able to access several pieces of sensitive data, such as: the data given as input by the user invoking the composite service (hereafter credentials), the final outcome of the composite service, as well as internal parameters (i.e., variables and values generated as output by partner services).

To cope with such issues, in this chapter we present our proposed secure protocol based on the selective encryption of user credentials and service parameters. Our protocol ensures that the broker is not able to access these data as well as any information on invoked activities, whereas partner services are able to access only the portions of user credentials and service parameters generated by other partner services, needed for the correct execution of the assigned workflow activity. The adopted encryption scheme allows the broker to evaluate test conditions on encrypted data, which is instrumental to enforce the correct execution order.

The rest of the chapter is organized as follows. Section 5.1 discusses the security requirements for the service composition, whereas Section 5.2 introduces the proposed architecture. Sections 5.3 presents how to guarantee the order of a workflow execution in a secure way. Encryption schemes are described in Section 5.4 and 5.5. The secure test condition evaluation is presented in Section 5.6. Experiments are reported in Section 5.7, and security properties of the proposal are described in Section 5.8.

5.1 Security Requirements for Service Composition on Untrusted Broker

In order to discuss the security requirements of composite web services, let us introduce a composite service for an e-commerce transaction.

Example 10. Let us consider an e-commerce transaction in e-shop E1 which has established a business partnership with two other e-shops, namely, E2 and E3. In case E1 is temporarily out of stock for an item, E1 re-directs the interested customers to the other e-shops. In terms of web services, the business process implies the composition of five different services, that is, one for each e-shop, plus the payment service and the delivery service. The BPEL document encoding this workflow is represented in Figure 5.1. As depicted in the figure, the first service to be invoked is the *login* operation from E1. This step is iterated until the login succeeds. The next invoked service aims at verifying if the requested item is available at E1 (line 6). If it is available, that is, the *rescheck* parameter is true, the BPEL engine invokes the payment service, by passing the service parameters, i.e. *itemID*, *price* and user credential *credit card* (line 17). In case it is not available, the BPEL engine invokes service E2 (line 8). It then proceeds for the payment (line 10), if the item is available. Then, the item is delivered by invoking service Delivery (line 11). Again, if the item is not available in E2, the BPEL engine invokes service E3, then the payment service, and, eventually, the delivery service.

This composition highlights that, even in a simple workflow, several pieces of user data are exposed to services as well as to the broker (for instance, *user name*, *password*, and *credit*

```
1.  <sequence>
2.  <invoke partnerLink="E1" operation="login" input="username" input="password" output=reslogin
    />
3.  <while> <condition>$reslogin="No"</condition>
4.      <invoke partnerLink="E1" operation="login" input="username" input="password"
    output=reslogin />
5.  </while>
6.  <invoke partnerLink="E1" operation="checkitem" input="itemID" output=rescheck output=price />
7.  <if> <condition>$rescheck="No"</condition>
8.  <invoke partnerLink="E2" operation="checkitem" input="itemID" output=rescheck   output=price />
9.    <if> <condition>$rescheck="Yes" </condition>
10.      <invoke partnerLink="Payment" operation="pay"  input="itemID" input="creditcard"
    input="price" output=receipt />
11.      <invoke partnerLink="Delivery" operation="deliver" input="itemID" input="receipt"
    output=status  />
12.   <elseif>
13.   <invoke partnerLink="E3" operation="checkitem" input="itemID" output=rescheck  output=price
    />
13.      <if> <condition>$rescheck="Yes"</condition>
14.       <invoke partnerLink="Payment" operation="pay" input="itemID" input="creditcard"
    input="price" output=receipt />
15.       <invoke partnerLink="Delivery" operation="deliver" input="itemID" input="receipt"
    output=status />
16.       </if>   </elseif>   </if>   <elseif>
17.   <invoke partnerLink="Payment" operation="pay" input="itemID" input="creditcard" input =
    "itemID" input="price"    output=receipt />
18.  <invoke partnerLink="Delivery" operation="deliver" input="itemID" input="receipt" output=status
    />
19.  </elseif>   </if>   </sequence>
```

Fig. 5.1 An example of BPEL document

card information). Similar concerns can be raised on the service parameters as well. [1] In the previous example, partner services might prefer to keep the *price* parameter confidential (lines 10, 14, 17). This problem is further exacerbated if we consider that the *broker might be untrusted*, in the sense that it could maliciously access and use user credentials and service parameters, as well as, it might be subject of attackers who take control over user data. To cope with these issues, the first relevant requirement for a secure web service composition is to ensure that *each partner service is able to access only the portions of user credentials and service parameters that it needs to complete its task, whereas the broker cannot access any of them*.

It is relevant to note that some of the output parameters are needed by the broker (i.e., BPEL engine) to evaluate conditions in the *if/while* instructions contained in the BPEL process. In Example 10, this is the case of *reslogin* and *rescheck* parameters (see lines 3, 7, 9, 13 in Figure 5.1). Therefore, a further requirement is that *the broker has to be able to evaluate conditions on certain output parameters by, at the same time, not being able to access their real values.*

Under the scenario of an untrusted broker, we see an additional requirement to satisfy, related to the risk that a malicious broker invokes services in an incorrect order. This might create a damage for users. For instance, in the composite service of Example 10 a malicious broker could invoke the *Payment* service without having checked before the item availability. As a result, the payment charges the user even if the item is not available. For this reason, a further requirement is that *the partner services have to be invoked and access their input parameters only according to the correct execution of the assigned workflow activity in the BPEL document*.

5.2 An Architecture for Secure Web Service Composition with Untrusted Broker

In order to enforce the above described security requirements, we design the *WSApp* application (see Figure 5.2), that makes a user able to invoke a composite service on an untrusted broker. As depicted in Figure 5.2, the user describes his/her requirements for composite services through a GUI. These are passed to an external web service, say *WFModeler*, which retrieves a suitable workflow from libraries of well-known business process patterns. The retrieved workflow is then sent to the *BPEL Generator* (step 2 in Figure 5.2). Based on

[1] Notice that there are two different kinds of parameters, that is, input parameters and output parameters. Input parameters are sent to partner web services and used as input for their activities; output parameters are values released from a partner web service.

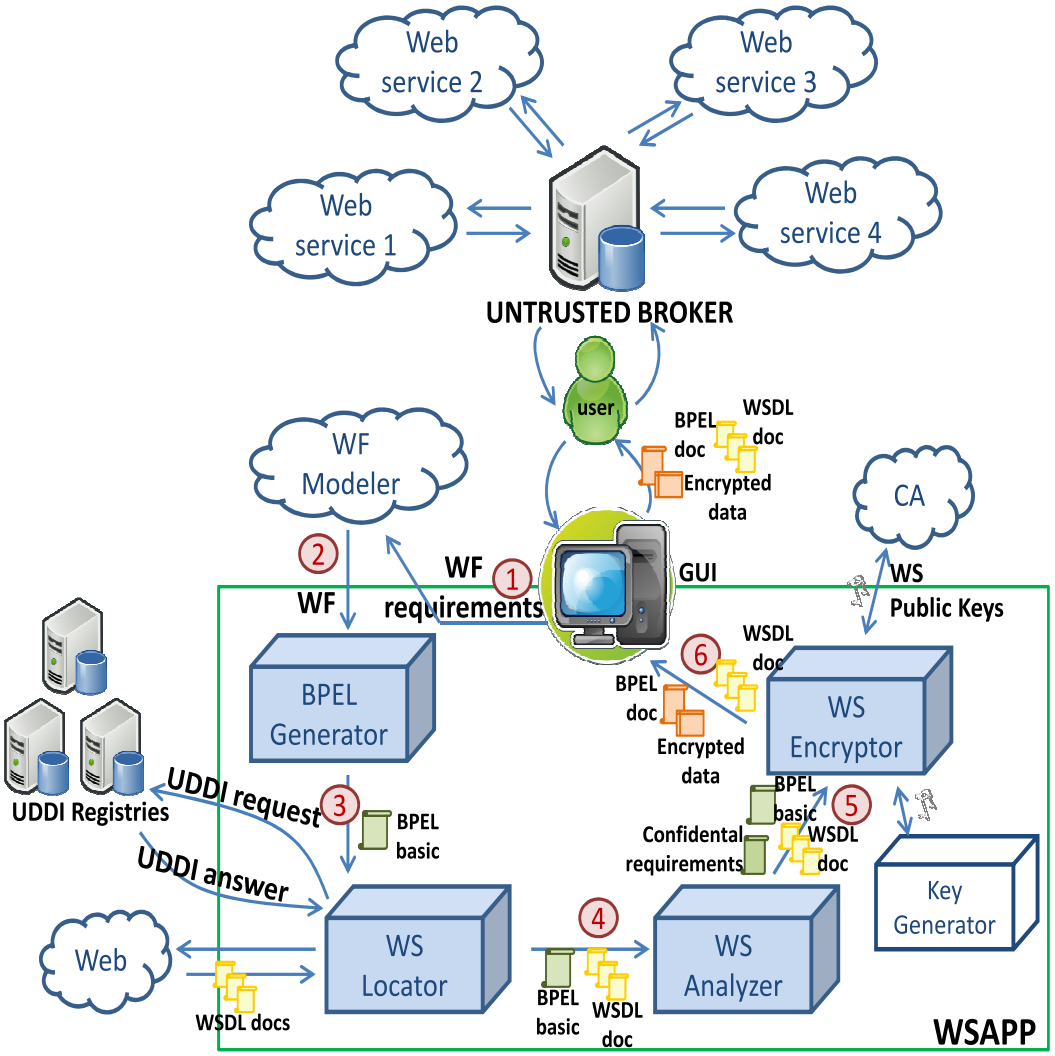


Fig. 5.2 An Architecture for Secure Web Service Composition

the specification of the received workflow, the *BPEL Generator* generates a basic BPEL document encoding the workflow, along with the description of input/output parameters, needed services, and their to-be-called operations. The obtained document is passed to the *WS Locator*. For each workflow activity, the *WS Locator* retrieves a service with functionalities requested for executing the workflow activities, by using UDDI search functionalities and semantic annotations. Additionally, the *WS Locator* retrieves *WSDL* documents specifying published interfaces for each web service. The *WS Locator* sends the BPEL and WSDL documents to the *WS Analyzer* (step 4 in Figure 5.2). This component identifies which portions of user credentials and output parameters are needed by each partner service. The final component of *WSApp* is the *WS Encryptor*, which generates the necessary encrypted data. In particular, it retrieves from a *Certificate Authority* (CA) the public keys of all partner services involved in the composition. It also requests the *Key Generator* module to create a set of secret keys. As it will be explained in the following sections, these keys are used for generating a secure version of the workflow description to which we refer hereafter as *encrypted BPEL document*. This is sent to the broker for its execution (step 6 in Figure 5.2).

5.3 Secure Workflow Execution

BPEL offers several types of activities to coordinate the service composition. Some of them contain sensitive information, e.g., user credentials, service parameters. In a scenario of untrusted broker, our proposal aims at hiding all this sensitive information. In particular, we focus on the invoked activity, which encodes information needed by the broker to invoke the required service. This can be a *direct invocation*, like the one in line 2 in Figure 5.1, or a *condition-based invocation*, that is, an invocation made after the BPEL engine has tested some conditions, as line 3 in Figure 5.1. In this chapter, we focus on condition-based invocation within *if* and *while* instructions. In order to hide sensitive information, we make the broker able to process an *encrypted BPEL document*, by enforcing, at the same time, the original workflow. At this purpose, for each invocation, the document contains an *encrypted activity block* (EAB). For a direct invocation, EAB consists of two components. As depicted in Figure 5.3, the first component, called *service label*, contains the minimum set of information needed by the broker for service invocation (e.g., the *partnerLink* information). The second component, called *encrypted activity description* (EAD), encrypts the description of the activity to be invoked and complementary information. The latter component has to be processed directly by the partner services, only when needed. At this purpose, the EAD is encrypted with the public key of the service that has to process it.

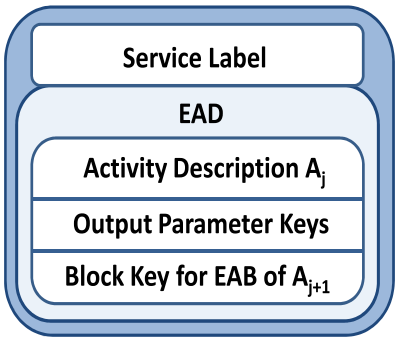


Fig. 5.3 Encrypted Activity Block for direct invocation

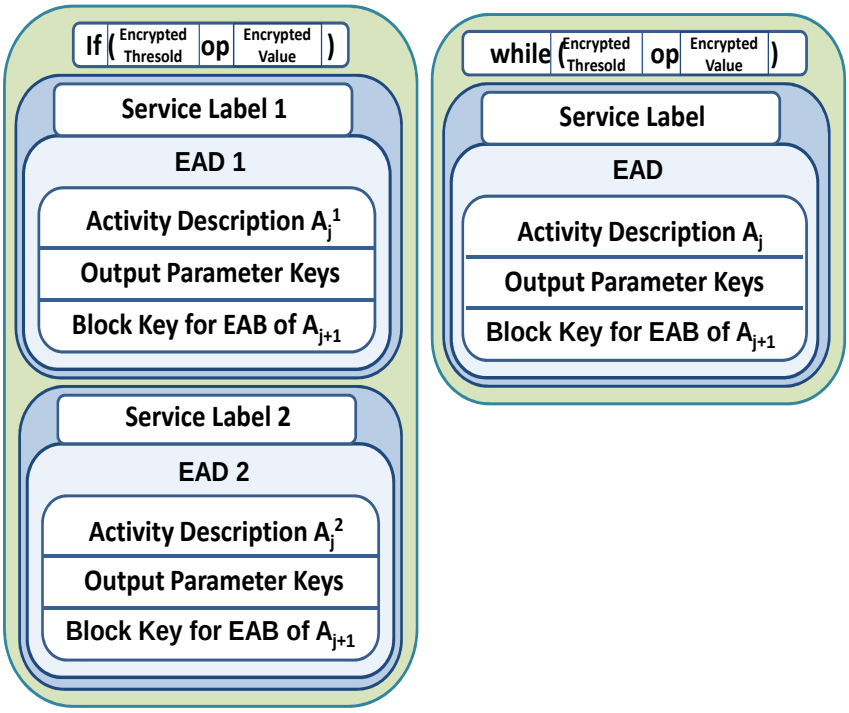


Fig. 5.4 Encrypted Activity Block for condition-based invocation

For condition-based invocations, EAB is a bit more complicated since the encryption that contains the information for making the broker able to securely evaluate test conditions. Moreover, the activity to be invoked depends on the result of the condition evaluation. To cope with these issues, EAB for condition-based invocations includes the information needed for the secure comparison and the nested EABs one for each possible activity to be invoked. As an example, in case of *if* activity, the condition-based EAB contains in turn two distinct EABs (see Figure 5.4).

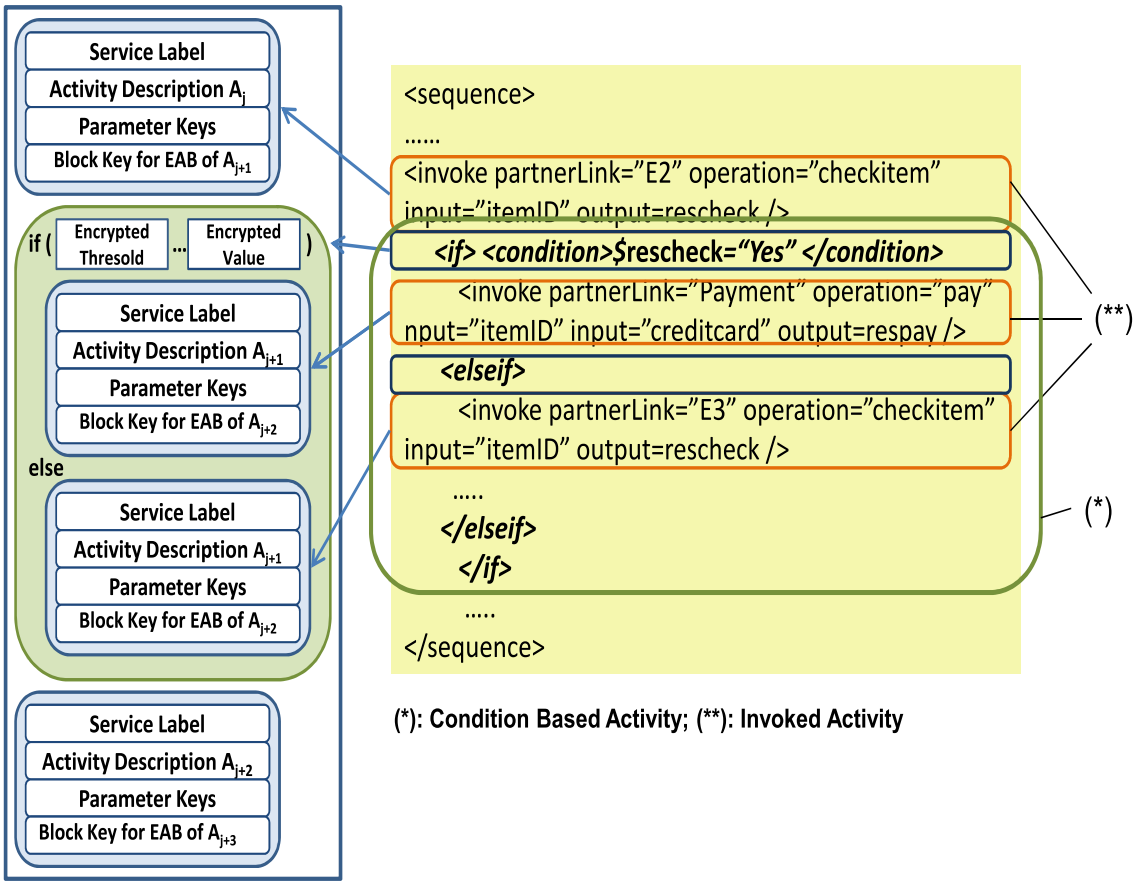


Fig. 5.5 Structure of an encrypted BPEL document

Each EAB is encrypted by a secret key generated by *WSApp*, called the *block key*. To ensure the correct execution order, the block key has to be available to the broker only when the previously invoked service has terminated its execution. At this purpose, given an activity A_{j+1} the corresponding block key $BK_{A_{j+1}}$ is embedded into the EAB of A_j (see Figure 5.3). Then, when the service carrying out activity A_j finishes it, the service sends its encrypted output parameters (see section 5.4 and 5.5) along with the block key $BK_{A_{j+1}}$ to the broker. Thus, the broker can use the block key $BK_{A_{j+1}}$ to identify the service to be invoked.

In particular, in case of direct invocation activity (see Figure 5.3), this is done according to the following steps: (1) it decrypts the next EAB to be processed, (2) it extracts service information from the service label, (3) it invokes that service by passing the EAD extracted by the encrypted block with enclosed encrypted input parameters for that service. In case of condition-based invocation activity (see Figure 5.4), the broker has to evaluate the condition according to a secure comparison scheme. As it will be described in Section 5.6, this evaluation will return the block key that has to be used for decrypting the proper nested EAB. That decryption also allows the broker to learn what service has to be invoked. In case of *while* activity the block key is not enough. Indeed, if the service invoked in the *while* cycle sends the block key for the next EAB to the broker, this could use it even if the test condition is not satisfied. Obviously, this breaks the requirement of correct workflow execution. To avoid this problem, for *while* activity the block key in the nested block is set to *null*. After testing the condition stated in the condition-based invocation, the broker makes a secure query (described in Section 5.6) to get a shared key for decrypting the nested block.

In the following, before presenting the formal definition of EAB (cfr. Section 5.5), we introduce the selective encryption of user credentials and service parameters.

5.4 Selective User Credentials and Parameters Encryption

In order to assure the confidentiality requirements discussed in Section 5.1, we propose to selectively encrypt user credentials and service parameters. Let us start by considering user credentials.

Definition 4. *Confidentiality Requirements on User Credentials.* Let u be the user requiring the composition of a service S, where u credential is represented as $C_u = \{att_1, att_2, \ldots, att_n\}$, with $att_j = (name_j, value_j)$, $j \in [1,n]$. Let WF_S be the workflow encoding the business process for S, and $WS = \{ws_1, ws_2, \ldots, ws_m\}$ be the set of web services assigned to activities in WF_S. For each $att_j \in C_u$, we represent its confidentiality requirements returned by the *WS Analyzer* as $Req_{att_j} = (att_j, \overline{ws})$, where $\overline{ws} \subseteq WS$.

Example 11. Let us consider the composition of Example 10, where the set of services is: $WS = \{E1, E2, E3, Payment, Delivery\}$. Here we assume that the customer, having username u and password Y, bought a shirt by using the *creditcard* with the account number X. u credential is $C_u = \{(username, u), (password, Y), (creditcard, X)\}$. According to the workflow modeled in the BPEL document in Figure 5.1, the user confidentiality requirements state that *username* and *password* can be only accessed by service E1, *creditcard* only by service Payment. As such: $Req_{username} = \{username, \{E1\}\}$, $Req_{password} = \{password, \{E1\}\}$, $Req_{creditcard} = \{creditcard, \{Payment\}\}$.

We therefore propose the following encryption for user credentials.

Definition 5. *User Credentials Encryption.* Let $C_u = \{att_1, att_2, \ldots, att_n\}$ be a user credential. Let $Req_{att_j} = (att_j, \overline{ws})$ be the confidentiality requirements on att_j, $j \in [1, n]$. The encryption of att_j is defined as a set $E_{att_j} = \{(Enc_{Pk_{ws}}(att_j.name), Enc_{Pk_{ws}}(att_j.value)) \mid \forall ws \in \overline{ws}\}$, where Pk_{ws} denotes the public key associated with service ws and $Enc()$ is an asymmetric encryption function.[2] The encryption of C_u, denoted as E_{C_u}, is $E_{C_u} = \{E_{att_j} \mid \forall att_j \in C_u\}$.

Example 12. Based on Example 11, let PK_{E1}, $PK_{Payment}$ be the public keys of $E1$ and $E_{Payment}$ respectively. Let C_u be u's credential. Then, $E_{C_u} = \{(Enc_{Pk_{E1}}(username), Enc_{Pk_{E1}}(u)), (Enc_{Pk_{E1}}(password), Enc_{Pk_{E1}}(Y)), (Enc_{Pk_{Payment}}(creditcard), Enc_{Pk_{Payment}}(X))\}$

While the encryption of user credentials can be generated by the *WS Encryptor* at *WSApp* side, the encryption of service parameters has to be performed by the service who has released them. At this aim, each service should know which are the services that will have to access its output parameters, but this is not always possible. This is the case, for instance, of an *if* activity, where the service to be invoked is not known at the time of BPEL definition. To cope with this problem, we propose to encrypt the relevant output parameters, with a different secret key. This key, hereafter denoted as *shared key*, is generated by *WSApp* (i.e., *Key Generator*) and stored in a tuple of the dataset, say *SecDB* (as introduced more detailed in Section 5.6), to be sent to broker. To avoid the broker to misuse these shared keys, they are encrypted with the public key of the partner service selected based on the result of the test condition evaluation. More precisely, let P be an output parameter and $\{ws_1, \ldots, ws_n\}$ be the set of services that, based on the result of the condition evaluation, could be authorized to consume P. For each ws_j, $j \in \{1, \ldots, n\}$, the *WS Encryptor* generates a different encryption of the shared key for P, say SK_P, that is, $Enc_{PK_{ws_j}}(SK_P)$. As it will be described in the next section, these encryptions of shared keys are included into the EAB for condition-based invocation.

[2] we use the dot notation to refer to components of a credential.

There is also another issue related to the fact that an output parameter from a prior invoked partner service is likely to be used as a comparison value in the next test condition instruction. For example, the output parameter *rescheck* from *E1* (see line 6 in Figure 5.1) is used as a comparison value in the test condition instruction (line 7). However, it is compulsory that the *rescheck* cannot be read by any unrelated partner services and the broker. So it is necessary to encrypt it. Hence, for each activity A_i, *WSApp* generates a pair of ECC public and private keys. More precisely, since the scheme uses only the public key, for sake of clarity, hereafter, we refer only to this key as the *Condition Based Key* for activity A_i, denoted as CB_{A_i}. For example, the output parameter *rescheck* from *E1* that is used for the next condition-based activity A_i will be encrypted with CB_{A_i} and becomes $Enc_{CB_{A_i}}(rescheck)$.

5.5 Encrypted Activity Blocks

Let us start to present the EAB for direct invocation, by first introducing a formalization of this kind of activity.

Definition 6. *Direct Invocation Activity.* We represent a direct invocation activity as tuple $A = (service\ address, operator, input\ parameters,$
$output\ parameters)$, where *service address* is the address of the service to be invoked, $operator$ is the operation performed by the service to be invoked, $input\ parameters$ are the required values for $operator$, and $output\ parameters$ are the values returned from the $operator$.

Let us consider the invocation activity A in line 2 of Figure 5.1. It can be modelled as $A = (E1, login, \{username, password\}, \{reslogin\})$.

Definition 7. *EAB for Direct Invocation.* Let A_i be a direct invocation activity, and ws_i be the service assigned to A_i. The EAB for A_i is defined as the encryption with the block key associated with A_i, BK_{A_i}, of the following components: 1) SL=A_i.*service_address*, that contains the address of the service to be invoked;[3] 2) EAD_{A_i}, that is, the encrypted description of activity A_i (as mentioned in Section 5.3) with the public key of the partner web service, i.e. PK_{ws_i}. EAD_{A_i} includes the following components: $ActDes_{A_i}$, that is, the BPEL code of A_i; OPK_{A_i}, which contains the keys used for encrypting output parameters, that is, (1) the public keys of the next direct invocation activity, i.e. $PK_{ws_{i+1}}$, and/or (2) the condition-based keys, i.e. $CB_{A_{i+1}}$, and/or (3) the shared keys, i.e. $SK_{A_{i+1}}$, of the next condition-based activity; NBK_{A_i}=$Enc_{PK_{broker}}(BK_{A_{i+1}})$, that is, the encryption of the block key of the next EAB with the broker public key.

[3] Hereafter, we adopt the dot notation to refer to components of an activity.

Thus, $EAB_{A_i} = Enc_{BK_{A_i}}(SL, EAD_{A_i}) = Enc_{BK_{A_i}}(SL, Enc_{PK_{ws_i}}(ActDes_{A_i}, OPK_{A_i}, NBK_{A_i}))$.

Example 13. With respect to the BPEL document in Figure 5.1, once the user completes the payment, he/she selects the delivery method by means of service Delivery. Payment's output parameter, i.e., receipt, is sent to service Delivery (line 11). Delivery also receives itemID from E1. Let A_i and A_{i+1} be the activities assigned to service Payment and Delivery, respectively. Hence, we have that EAB for activity A_i has the following components: $SL_{A_i} = Payment$; $ActDes_{A_i}$= "$< invoke\ partnerLink = Payment\ operation = pay\ input = creditcard\ input = itemID\ input = price\ output = receipt / >$"; OPK_{A_i}=$\{PK_{Delivery}\}$, where $PK_{Delivery}$ is the public key of Delivery that has to be used for encrypting of the parameter *receipt* for the next activity Delivery; $NBK_{A_i} = Enc_{PK_{broker}}(BK_{A_{i+1}})$. Hence, $EAD_{A_i} = Enc_{PK_{Payment}}(ActDes_{A_i}, OPK_{A_i}, NBK_{A_i})$ and $EAB_{A_i} = Enc_{BK_{A_i}}(SL_{A_i}, EAD_{A_i})$.

As highlighted in Section 5.3, condition-based invocation requires to include into the corresponding EAB more information. Again, let us start with a formalization of this kind of activity. More precisely, we consider the if and while activity.

Definition 8. *If Activity.* We represent an if activity as a tuple $A = (value, threshold, inv\ activity\ 1, inv\ activity\ 2)$, where *value* is the value returned as output parameter from a previous activity that has to be compared with $threshold$, and *inv activity* 1, *inv activity* 2 are invoke activities formalized according to Definition 6. If $value = threshold$, then *inv activity* 1 is enforced; otherwise, *inv activity* 2 is enforced.

Definition 9. *While Activity.* We represent a while activity as a tuple $A = (value, threshold, inv\ activity)$, where *value* is the value returned as output parameter from a previous activity that has to be compared with $threshold$, and *inv activity* is an invoke activity formalized according to Definition 6. Until *value* is not equal to $threshold$, the *inv activity* is enforced.

Let us see the corresponding EAB. In both if and while condition-based activity A_i, the broker has to evaluate the condition $A_i.value = A_i.threshold$. Value is an output parameter returned by a previously invoked service ws_j, $j < i$. As introduced in Section 5.3, in case of if activity, the corresponding EAB contains two nested EABs for the services to be invoked. These are encrypted with two separate block keys that are retrieved by the broker as the result of a query on *SecDB*. With each condition-based invocation *WsApp* associates a different public key, denoted as CB_{A_i} (as it will be introduced in Section 5.4).

Definition 10. *EAB for If Activity.* Let A_i be an If activity. Let A_{i_1}, A_{i_2} be the two nested invoked activities in A_i. Let $BK_{A_{i_1}}$, $BK_{A_{i_2}}$ be the shared keys used for generating EABs of A_{i_1}, A_{i_2}, respectively. Let CB_{A_i} be the condition-based key associated with A_i. The

EAB for A_i is defined as the encryption with BK_{A_i} of the following components: 1) $ET = Enc_{CB_{A_i}}(A_i.threshold) \oplus Enc_{CB_{A_i}}(\phi_{A_i})$, where ϕ_{A_i} is the unique random number associated with A_i, and is also the primary key in case the condition instruction is satisfied (see more detailes in Section 5.6); 2) $EV = Enc_{CB_{A_i}}(A_i.value)$, that is, the encryption with CB_{A_i} of the value to be compared with *threshold*. This is returned as an output parameter by a previously invoked service; 3) $IA1 = Enc_{BK_{A_{i_1}}}(SL1, EAD_{A_{i_1}})$, as defined in Definition 7; 4) $IA2 = Enc_{BK_{A_{i_2}}}(SL2, EAD_{A_{i_2}})$, as defined in Definition 7.

Thus, $EAB_{A_i} = Enc_{BK_{A_i}}(ET, EV, IA_1, IA_2)$.

Example 14. Let A_i be the if activity in line 7 of Figure 5.1. The components of the corresponding EAB are defined as follows: $ET = Enc_{CB_{A_i}}(\text{"}No\text{"}) \oplus Enc_{CB_{A_i}}(\phi_{A_i})$; $EV = Enc_{CB_{A_i}}(rescheck)$; $IA_1 = Enc_{BK_{A_{i_1}}}(SL1, EAD_{A_{i_1}})$, where $SL1 = \text{"}E2\text{"}$, $EAD_{A_{i_1}} = Enc_{PK_{E2}}(ActDes1, OPK1, NBK1)$ with $ActDes1 = (\text{"} < invokepartnerLink = E2\ operation = checkitem\ input = itemID\ output = rescheck\ output = price/ > \text{"})$, $OPK1 = \{CB_{A_{i+1}}, PK_{Payment}\}$, $NBK1 = Enc_{PK_{broker}}(BK_{A_{i+1}})$, $BK_{A_{i+1}}$ is the block key of the next EAB; $IA_2 = Enc_{BK_{A_{i_1}}}(SL2, EAD_{A_{i_1}})$ where $EAD_{A_{i_2}} = (ActDes2, OPK2, NBK2)$ with $SL = \text{"}E2\text{"}$, $ActDes2 = (Enc_{PK_{Payment}}(\text{"} < invoke\ partnerLink = Payment\ operation = pay\ input = creditcard\ input = itemID\ input = price\ output = receipt/ > \text{"})$; $OPK2 = \{PK_{Delivery}\})$, and $NBK = Enc_{PK_{broker}}(BK_{A_{i+1}})$. Hence, the encryption of A_i generated by the broker is: $Enc_{BK_{A_i}}(ET, EV, IA1, IA2)$.

Definition 11. *EAB for While Activity.* Let A_i be a While activity. Let A_{i_1} be the nested invoked activity in A_i. Let $BK_{A_{i_1}}$ be the block key used for encrypting A_{i_1}. Let CB_{A_i} be the condition-based key associated with A_i. The EAB for A_i is defined as the encryption with BK_{A_i} of the following components: 1) $ET = Enc_{CB_{A_i}}(A_i.threshold) \oplus Enc_{CB_{A_i}}(\phi_{A_i})$, where ϕ_{A_i} is the unique random number associated with A_i; 2) $EV = Enc_{CB_{A_i}}(A_i.value)$ is the encryption with CB_{A_i} of the value to be compared with *threshold*. *val* is returned as an output parameter by a previously invoked service; 3) m $WA = Enc_{BK_{A_{i_1}}}(SL_{A_{i_1}}, EAD_{A_{i_1}})$, where $EAD_{A_{i_1}} = (ActDes_{A_{i_1}}, OPK_{A_{i_1}}, NBK_{A_{i_1}})$ as defined in Definition 7. Thus, $EAB_{A_i} = Enc_{BK_{A_i}}(ET, EV, WA)$.

Example 15. Let A_i be the While activity specified in line 3 of Figure 5.1. The components of the corresponding EAB are defined as follows: $ET = Enc_{CB_{A_i}}(\text{"}No\text{"}) \oplus Enc_{CB_{A_i}}(\phi_{A_i})$; $EV = Enc_{CB_{A_i}}(reslogin)$; $WA = Enc_{BK_{A_{i_1}}}(SL_{A_{i_1}}, EAD_{A_{i_1}})$, where $SLA_{i_1} = \text{"}E1\text{"}$, and $EAD_{A_{i_1}} = Enc_{PK_{E1}}(ActDes_{A_{i_1}}, OPK_{A_{i_1}}, NBK_{A_{i_1}})$ with $ActDesc_{A_{i_1}} = \text{"} < invokepartnerLink = E1\ operation = login\ input = username\ input = password\ output = reslogin/ > \text{"}; OPK_{A_{i_1}} = \{CB_{A_i}\}$, and $NBK_{A_{i_1}} = null$. Hence, $EAB_{A_i} = Enc_{BK_{A_i}}(ET, EV, WA)$.

5.6 Secure Evaluation of Test Conditions

EAB for condition-based invocation has to contain information that makes the broker able to evaluate the condition without inferring the real values to be evaluated (e.g., the threshold and the parameter values). At this purpose, we make use of homormophic encryption, which allows to perform some simple math operations directly on encrypted data [122]. In particular, in this proposal, we initially investigate the '=' operator and we adopt the Elliptic Curve Cryptography algorithm [59] to exploit its additive homomorphism property. Let us start to introduce which information *WSApp* has to generate in order to support the secure comparison.

SecDB. We assume that the *Encryptor* generates a random number ϕ_{A_i} uniquely associated with activity A_i. We also assume that this module generates value $Enc_{CB_{A_i}}(A_i.threshold) \oplus Enc_{CB_{A_i}}(\phi_{A_i})$, which is enclosed into the EAB sent to the broker (see Section 5.5) where CB_{A_i} is a *Condition Based Key* as defined in Section 5.4.

For each condition-based activity A_i, two tuples are created by *WSApp* and stored into *SecDB*. These tuples have three components as follows:

- *Primary Key*. This component is used by the broker to retrieve the correct tuple to be elaborated. This tuple has to be selected based on the result of the test condition. Note that, according to the additive homomorphic ECC property, in case the condition is satisfied, i.e., $A_i.value = A_i.threshold$, if the broker computes the following exclusive-OR $Enc_{CB_{A_i}}(A_i.value) \oplus (Enc_{CB_{A_i}}(A_i.threshold) \oplus Enc_{CB_{A_i}}(\phi_{A_i})$, it obtains $Enc_{CB_{A_i}}(A_i.value \oplus A_i.threshold \oplus \phi_{A_i})$=$Enc_{CB_{A_i}}(\phi_{A_i})$; otherwise, the result is different. As such, we define the primary key of the tuple to be processed in case of the condition satisfied with the value $Enc_{CB_{A_i}}(\phi_{A_i})$. Thus, for each condition-based invocation activity, there are two tuples with the following indexes: (1) $index_1$=$Enc_{CB_{A_i}}(\phi_{A_i})$ for the tuple to be evaluated in case the condition is satisfied; and (2) $index_2$=$Enc_{CB_{A_i}}(A_i.threshold) \oplus Enc_{CB_{A_i}}(\phi_{A_i})$, for the tuple to be evaluated in case the condition is not satisfied.

- *Block Key*. The tuple has to contain the block key for the nested EAB. For each primary key of the same condition invocation activity as described in the first component, we have the respective block keys as follows:

(1) $index_1$=$Enc_{CB_{A_i}}(\phi_{A_i})$: the block key carries $BK_{A_{i_1}}$ of the next EAB. In case of If activity, it contains the $BK_{A_{i_1}}$ for EAB of the activity to be invoked if the condition is satisfied. In case of a While activity, it contains the $BK_{A_{i_1}}$ for EAB of the first activity to be invoked after the while cycle.

(2) $index_2$=$Enc_{CB_{A_i}}(A_i.threshold) \oplus Enc_{CB_{A_i}}(\phi_{A_i})$: in case of If activity, it contains the $BK_{A_{i_2}}$ for EAB of the activity to be invoked in case of not satisfied condition. In case of While activity, it contains the null value.

- *Encrypted Shared Keys*. This component contains the shared keys, i.e. $SK_{A_{i+1}}$, that have to be used to encrypt the output parameters of A_i invoked by ws_i, then used for A_{i+1} invoked by ws_{i+1}, in case A_{i+1} is a condition-based activity. In particular, this component contains the encryption of $SK_{A_{i+1}}$ with the public key of the service to be invoked. For each primary key of the same condition invocation activity, we have the respective block keys as followes:

(1) $index_1$=$Enc_{CB_{A_i}}(\phi_{A_i})$: the encryption of $SK_{A_{i+1}}$ with the public key of the service to be invoked in case the condition is satisfied. In case of If activity, the public key of the service invoked by *inv activity*1 (see Definition 10). In case of While activity, the encryption of $SK_{A_{i+1}}$ with the public key of the service to be invoked after the while cycle.

(2) $index_2$=$Enc_{CB_{A_i}}(A_i.threshold) \oplus Enc_{CB_{A_i}}(\phi_{A_i})$: the encryption of $SK_{A_{i+1}}$ with the public key of the service to be invoked in case the condition is not satisfied. In case of If activity, with the public key of service invoked by *inv activity*2 (see Definition 10). In case of While activity, it contains the *null* value.

Broker secure comparison process. When the broker has to evaluate the test condition in A_i, it performs the following steps: (1) from the EAB corresponding to A_i, it retrieves $Enc_{CB_{A_i}}(A_i.threshold) \oplus Enc_{CB_{A_i}}(\phi_{A_i})$ and $Enc_{CB_{A_i}}(A_i.value)$; (2) it computes the aggregate value, say AG, where AG= $Enc_{CB_{A_i}}(A_i.threshold) \oplus Enc_{CB_{A_i}}(\phi_{A_i}) \oplus Enc_{CB_{A_i}}(A_i.value)$; (3) it looks for a tuple in *SecDB* with primary key value equal to AG; in case this does not exist, it looks for a tuple with primary key equal to $Enc_{CB_{A_i}}(A_i.threshold) \oplus Enc_{CB_{A_i}}(\phi_{A_i})$; (4) it uses the block key contained in the tuple retrieved in step (3) to decrypt and elaborate the nested EAB. In case the tuple does not contain the block key (i.e, the while condition is not satisfied), it invokes again the last invoked service.

5.7 Experimental Results

In order to demonstrate the efficiency of the proposal, we carried out several experiments on the WSApp and broker prototype. We recall the we exploit asymmetric and symmetric encryption as well as and ECC. Therefore, in the prototype we implemented RSA (1024 bit and 2048 bit key size), AES (128 bit and 256 bit key size), ECC (163 bit, 283 bit, 409 bit and 571 bit key size). To estimate the overhead of WSApp, we measure the time needed to generate the encrypted BPEL documents, as well as the size of resulting encrypted BPEL documents. To evaluate the broker performance, we measure the time needed to elaborate each encrypted BPEL document. Since key size greatly impacts the performance of cryptographic algorithms, we carried out experiments with four different combinations of key sizes. These are represented in Table 5.1.

Table 5.1 Considered Key Sizes

ECC	RSA	AES
163 bit	1024 bit	128 bit
283 bit	1024 bit	128 bit
409 bit	2048 bit	256 bit
571 bit	2048 bit	256 bit

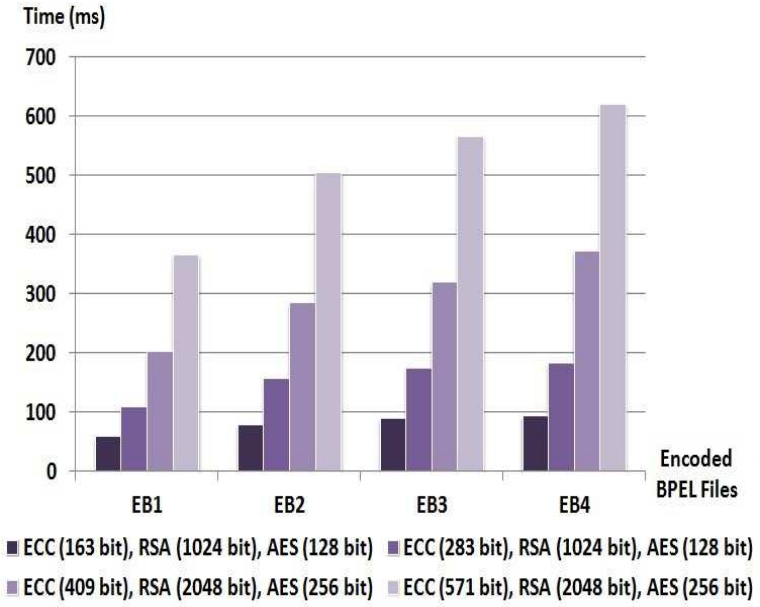
Each BPEL document has been encrypted about 50 times, and experimental results show the average overhead. All experiments have been deployed on a PC Intel Duo Core CPU 3GHz, 4GB RAM, and 64-bit Windows 7 Professional OS. We have used four different BPEL documents, say respectively B1, B2, B3, B4, each of which describes the collaboration among 25 services. Parameters of each original BPEL document are shown in Table 5.2.

Table 5.2 Parameters of Inpput BPEL Documents

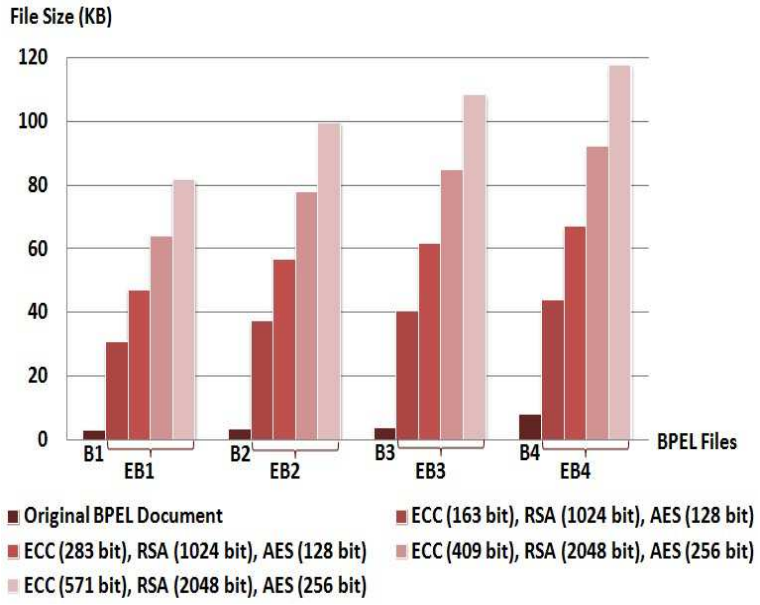
Original BPEL Documents	B1	B2	B3	B4
Number of Direct Invocation Activities	15	14	15	16
Number of Condition-Based Activities	7	10	11	12
Total Number of Activities	22	24	26	28
Number of Output Parameters	31	40	42	48
File Size (KBytes)	3	4	4	8

Figure 5.6 presents the performance of WSApp. Figure 5.6-a) shows the time needed to generate the encrypted BPEL documents. This time also includes the creation of SecDB tuples. As depicted in Figure 5.6-a), even in the worst case, that is, in case of the largest BPEL document and with the combination having the longest key size, the time of generation is not significant (i.e. under 1 second). Figure 5.6-b) shows the size of the obtained encrypted documents. Also in the worst case, the generated encrypted BPEL has the file size of 118 KBytes.

Figure 5.7 presents the time consumption for the broker to elaborate the encrypted BPEL documents. This does not include the time for service invocation and waiting for the response to/from the partner web services. The time is only for decrypting the encrypted BPEL document and making queries on *SecDB*. As reported in Figure 5.7, even in the worst case, the processing delay for reading the encrypted BPEL documents is minor, i.e. 1.39 seconds.



a) Time consumption (ms)



b) File size (KB)

Fig. 5.6 Time consumption and File size according to the number of activities and the number of output parameters at WSApp

5.8 Security Properties

In this section, we discuss the security properties of the proposed protocol. This discussion is done under the assumption that the broker is *honest-but-curious*, which implies that it correctly enforces the protocol, but it tries to retrieve as much information as possible. Let us

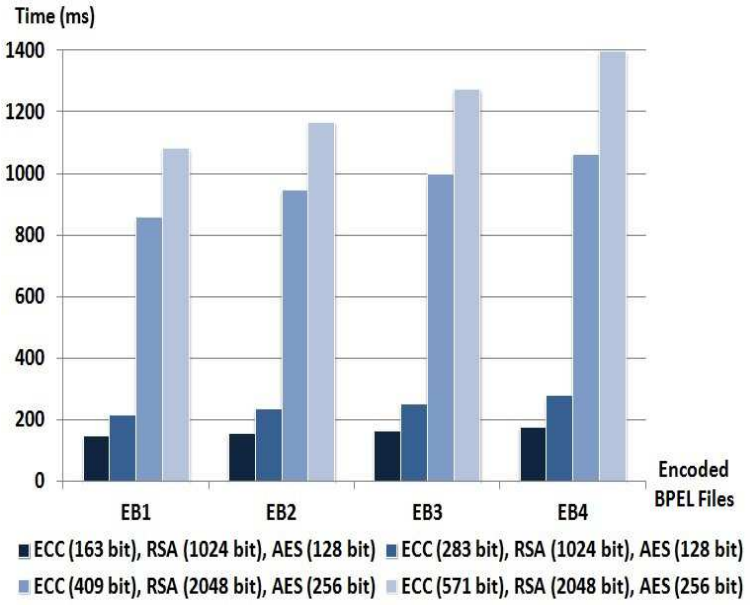


Fig. 5.7 Time consumption for reading the encrypted BPEL document according to the number of activities and the number of output parameters at the broker

consider each party in the protocol.

Broker. Let us consider each kind of data transferred through the broker from/to web services. In particular, according to the proposed protocols, these data are: (1) user credentials; (2) parameters in a direct invocation activity; (3) parameters in a test condition evaluation in an EAB; (4) parameters included into a nested EAB inside an EAB of a condition based activity; (5) activity description in an EAD.; (6) the data in support of secure test condition evaluation. Now let us clarify how the proposed protocol protects the above parameters and the test condition evaluation process. The broker cannot be aware of (1) since each user credential is encrypted by the RSA public keys of those web services authorized according to the BPEL document. Without the corresponding private keys, the broker is not able to decrypt it. Similarly to (1), parameters in (2) are encrypted by the RSA public keys of the authorized web services. Parameters in (3) are encrypted with the ECC public keys and used for the test condition evaluation. Since the broker does not possess any of the corresponding ECC private keys, it cannot access these data. Parameters in (4) are encrypted by AES shared keys, which are stored in *secDB* and encrypted by the public keys of the authorized web services, so the broker cannot decrypt them. Data in (5) are encrypted with the RSA public key of the authorized web service, so only authorized web service can read this activity description with its RSA private key. Regarding the process (6), the broker cannot obtain any relative information, because secure comparison is computed by exploiting

ECC homomorphic encryption, i.e. all data of the process are encrypted and the computation operators of the process are performed directly on those encrypted data.

Partner Web Services. A partner web service receives the encrypted activity description, user credentials and parameters from the broker. Regarding activity description and user credentials, the encryption scheme ensures that each partner service can decrypt only those authorized to it, that is, those encrypted with its public key. The same holds for parameters included in direct invocation activities. In contrast, parameters for condition-based activities are encrypted with a symmetric shared key. We recall that this key is in turn encrypted with the public key of web services authorized to access the parameters, based on evaluation of test condition. As such, the shared key and, as consequences, the output parameters can be decrypted only by authorized web services.

Chapter 6

A privacy-preserving constrained choreographed service composition

A Web service is a software system designed to support interoperable application-to-application interactions over the Internet. One of the major goals of Web services is to make easier their composition to form more complex services, modeled as workflows. As mentioned in Chapter I, there are two main techniques for web service composition, namely, orchestration and choreography. Choreography is a decentralized model with a dynamic sequence of web services, whereas, orchestration has a central broker to control a static sequence of activities. Literature shows that there is an increasing interests in decentralized composition as the orchestration paradigm suffers of single-point failure problem as well as requires powerful and reliable brokers. According to the choreography paradigm, once the business process and the workflow behind the web service composition has been defined, the composite service is deployed by invoking the service that has to perform the first activity. When this service correctly terminates the activity, it searches a service able to carry on the next activity in the workflow. This is then directly invoked by the service that has just terminated, without the presence of a broker (like in the orchestration model).

It is relevant to note that regardless the adopted paradigm, a crucial task is the selection of the service to be assigned to each activity in the workflow. As a matter of fact, service compositions can be driven by constraints on service selection. As discussed in [131], these requirements can be posed both by users requiring the composed service as well as by the companies provisioning the atomic services. In general, requirements can refer to several dimensions, such as Quality of Service (QoS) [86] or security [24], as services might have strong security requirements on services with which they have to cooperate during the service composition deployment. Literature presents several approaches for orchestrating constrained web service compositions, where QoS [94, 148, 128] and/or security criteria

[24] [131] have been considered . Decentralized approaches have been proposed as well [3, 151, 64]. However, to the best of our knowledge none of them considers that the evaluation of these criteria requires to expose private information of users and providers, regardless of whom poses the requirements (i.e., users or providers) and the nature of these requirements (e.g., QoS, security, etc.). The only paper we are aware of dealing with these issues is [132], where a centralized and privacy aware approach for service selection of composite services has been proposed. However, the peer-to-peer nature of the choreography model imposes to revise the way requirements have to be enforced. Indeed, the absences of a broker entity demands for *services able to locally and privately validate user and provider requirements* on new services to be invoked. At this purpose, in this chapter, we present our proposed a privacy-preserving framework for a choreographed service composition, where to enable privacy-preserving requirements evaluation we make use of two secure protocols.

The remainder of the chapter is organized as follows. Section 6.1 introduces privacy issues in the choreography of web service composition. The proposed framework is described in Section 6.2. Section 6.3 presents the user requirement evaluation, while Section 6.4 presents the provider requirement evaluation. Experimental results are reported in Section 6.5.

6.1 Privacy issues in a constrained choreographed web service composition

In order to illustrate the privacy issues we are interested in, let us assume that a user requests a composite service aiming at organizing a trip plan. Moreover, assume that the corresponding workflow implies the search and reservation of flight, hotel, car, and a tour visit. Based on user's preferences, each activity of the service composition can be further constrained by additional user-defined constraints. For example, the requesting user might pose conditions on flight budget (e.g., economy, business, premium economy), as well as food, seat position, frequent fliers, etc. All these additional search criteria are encoded into *user requirements*, whose formal definition is provided in Section 6.3.

According to choreography paradigm, the service deployment is then triggered by invoking the first service, i.e., the service carrying on the first activity, which, in turn, will search and invoke a service for the second activity in the workflow once it has terminated its own activity. When invoking the second service, all the information about workflow as well as users credentials is passed, so as to make the new service able to invoke the successive one. This process is continued until the workflow completion. This is the general way a choreographed web service composition is carried out.

We have to note that, this peer-to-peer nature of the choreography model imposes to revise the way user requirements have to be enforced. Indeed, the absences of a broker entity demands for *services able to locally validate user requirements on new services to be invoked.* In order to deal with this scenario, we assume that each service is characterized by a set of *properties*, defined as pair of property name and corresponding value(s). Thus, evaluating user's requirements implies to evaluate a set of constraints on properties of the service to be invoked. This obviously implies to expose to the service who has to locally evaluate these conditions the users' private information (i.e., user preferences) as well as parameter values of service to be invoked. Some of these data might be very sensitive, as an example user's preference on food might reveal the user's religion. Moreover, the service provider might have concerns to expose its property values to an unknown service. As an example, the user's constraint on budget requires the service to expose its price to previously invoked services, i.e., the ones that have to evaluate user's requirements. Thus, the privacy-preserving framework has to ensure that: *(R1) user's requirements have to be privately evaluated over service properties without revealing to the service carrying on the evaluation any of the requirement-related information (e.g., parameters' values, comparison values, etc.).*

It should be also taken into account that, during a service composition, providers might have requirements on the other services with which they have to cooperate. *Provider requirements* might be related to security [24, 131], imposing, as an example, to cooperate only with services adopting appropriate security mechanisms. For instance, the flight reservation service might require to invoke only services adopting a given encryption algorithm (e.g., AES with 256 bits key size). Moreover, QoS related constraints might be posed as well (e.g., response time below a given threshold). It is relevant to note that providers at both the sides of the composition might have requirements, that to say, both provider of the invoking service as well as of invoked service might have its own requirement on service (to be invoked or from which being invoked).[1]

Similarly to the user requirements enforcement, in the choreography scenario each service has to locally evaluate its constraints on the next service properties. For instance, the flight reservation service has to evaluate whether the hotel booking service adopts the required encryption algorithm. Again, even for provider requirements, we see privacy issues as evaluating these constraints implies to expose confidential information. As such, *(R2) provider requirements have to privately evaluated over service properties without revealing any of the requirement-related information (e.g., parameters, comparison values, etc.).*

[1]The approach we present in this chapter for privacy-preserving evaluation of providers requirement can be used for both these evaluations. Without lacks of generality, in the following we focus on requirements of the invoking services.

Further, we have to note that delegating the evaluation of user and providers requirements to each single service might bring to harmful situations, as the delegated service as well the service providing the properties might cheat the system. This can be done in several ways. As an example, by skipping the evaluation or by unfairly evaluate conditions on fake property values so as to invoke a colluding service rather the one satisfying the considered requirements. In order to prevent this scenario, the proposed framework has to *(R3) provide proofs of the correct privacy-preserving evaluation of user and providers requirements*.

6.2 A privacy-preserving framework for choreographed composition

To cope with the requirements introduced in Section 6.1, we propose a solution where each service is able to locally evaluate user and providers requirements in a privacy-preserving way. This implies that each service has to privately evaluate conditions over user credentials and service provider properties. To enable this private evaluation we make use of two secure protocols proposed in [16] and [45]. This requires to initialize some security-related parameters and data structures. At this purpose, we propose to include in the service composition choreography an additional component, called ElitePicker application (EPApp).

As depicted in Figure 6.1, EPApp is required only during the initialization phase. Thus, a user wishing to deploy a composite service has to first interact with EPApp. EPApp gathers the user service request, e.g., a trip organization. Then, it deploys the workflow that in the context of choreography is intended as a description of activities, which each services has to carry on, and a description of the coordinated interactions among services. For this workflow specification, we adopt WS-CDL (Web service choreography description language) [125] to dictate the order in which services have to collaborate according to a defined plan. However, WS-CDL is not an executable business process description language, as well as, not used for exchanging data of requests/responses among web peers. So to let services able to invoke their required activities, WS-CDL can be used with one of executable business process languages, such as BPEL, WS-BPEL, etc. As well to exchange request/response data, we use SOAP [150]. Moreover, in this chapter, we model sequential workflows, in the form $WF=\{a_1, a_2, \ldots, a_{n-1}, a_n\}$.

Through EPApp's GUI, the user specifies also his/her requirements, according to the definition given in Section 6.3. In general, applying a user requirement to a workflow activity a_j implies that properties of service that has to carry on a_j have to satisfy the corresponding conditions. In particular, in order to satisfy $R1$, we adopt the protocol proposed in [16], which

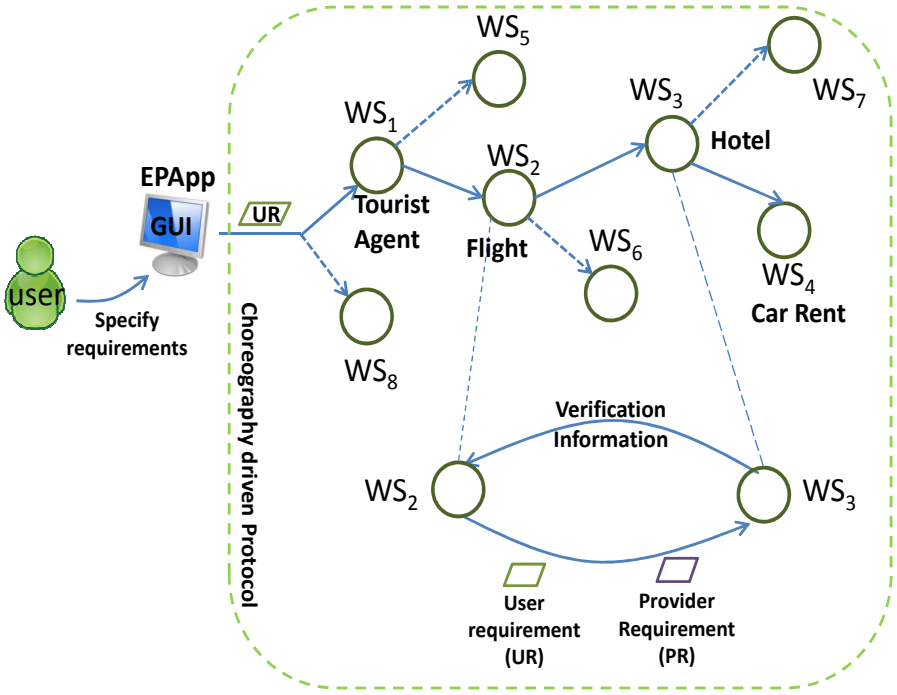


Fig. 6.1 An example of web service selection

exploits asymmetric encryption to privately evaluate combination of predicates (see Section 6.3 for more details). Thus, we assume that, for each service request submitted by a user, EPApp generates a distinct pair of keys: a public key PK, available to everyone, and a secret key SK, which is held only by the user requiring the composition. Then, the secret key is used by the requestor to generate a distinct data structure, called *token*, for each of his/her requirements. This data structure contains the encryption of the information related to the user requirement (e.g., parameters, threshold values). As it will be described in Section 6.3, a token is defined in such a way that only the service carrying on the activity to which the user requirement applies is able to decrypt and securely evaluate it over its property values. As an example, the user requirement asking for a flight with price less than 100 has to be evaluated only by the flight booking service.

As depicted in Figure 6.1, to start the composition, by means of EPApp, the user searches for service WS_1 which has to execute the first activity a_1.[2] Afterward, EPApp passes all the generated tokens along with user PK to WS_1. In turn, WS_1 retrieves the token(s) applying to activity a_1 and, following the protocol in [16], it privately evaluates whether its properties

[2]This service is retrieved by inquiring Universal Description, Discovery and Integration (UDDI) as well as other search methods.

satisfy the stated conditions. As it will be described in Section 6.3, it generates a proof that can be used by EPApp to check whether the requirement evaluation has been correctly executed. If this is the case, EPApp invokes WS_1. A similar process is done for each activity a_j in the workflow. Therefore, once activity a_{j-1} has been executed, the corresponding service searches for a new service WS_j to be assigned with a_j, it sends to it all tokens along with the user PK. WS_j will be then invoked only if WS_{j-1} receives a proof that WS_j satisfies the user requirements.

As introduced in Section 6.1, in addition to user requirements, we have to deal with provider requirements as well. For their privacy-preserving evaluation, it is not possible to exploit the protocol adopted for user requirements. The main reason is that provider requirements are not known during the initialization phase. Indeed, the dynamic selection of services to be assigned to each activity makes impossible to predict their requirements. To deal with this issue, we adopt the privacy preserving matching protocol presented in [45], which aims at making two entities able to privately compute the intersection of two sets. The key idea is that, to check if a given value satisfies a condition, we can verify whether it belongs to the predefined set of values satisfying the condition. As an example, given the condition “ExpectedTimeResponse$<$10" the value 9 does satisfy the condition as it belongs to the set [0,9). Thus, as it will be described in Section 6.4, we exploit the protocol defined in [45] to make the invoking service and the service to be invoked able to privately verify whether the requirements of the invoking service are satisfied. More precisely, when a service carrying on the generic activity a_j ended its execution, it first verifies user requirements on WS_{j+1}, i.e., the candidate service for activity a_{j+1}. Then, it verifies if WS_{j+1} is compatible with its requirements. Also in this case, the protocol execution generates a proof that can be used to check the correct evaluation. The protocol is presented in details in Section 6.4.

As a final note, in order to satisfy requirement $R3$, all proofs generated during the composite service deployment are collected and returned to EPApp together with the final results. By exploiting these proofs, EPApp can further verify that all privacy-preserving evaluations have been done correctly, that is, that the services evaluating the requirements have not colluded with the service under evaluation.

6.3 Privacy-preserving user requirements evaluation

When a user u submits to EPApp a request for a composite web service, EPApp returns the workflow WF underlying the composition. Then, u specifies, for each activity $a_i \in WF$, the local requirements that have to be satisfied by the web service that will carry on this activity. In general, user requirements are defined as boolean expressions of atomic conditions.

Atomic conditions pose constraints on values of a service property. We assume that services adopt the same catalog of property names, $\mathscr{P}$, defined according to a common ontology.[3]

Definition 12. *Atomic condition.* Let $\mathscr{P}$ be the set of property names. An atomic condition *cond* on $\mathscr{P}$ is defined as: $cond = [prop, op, thresh]$, where $prop \in \mathscr{P}$ is a property name, $op \in \{<, >, \leq, \geq, =, \neq\}$ is a comparison operator, and *thresh* is a value in the domain of *prop*.

Example 16. In the trip organization scenario, the atomic condition requiring a price less than 50 Euro is modeled as $cond = [price, <, 50]$.

Based on the above definition, we model user requirements as follows.

Definition 13. *User requirements.* Let a_i be an activity specified in a workflow WF, modeling the composite web service required by user u. The requirements specified by u for a_i, referred to as UR_{a_i}, are defined as a Boolean expression over atomic conditions. We express UR_{a_i} in Disjunctive Normal Form [100] denoted as: $UR_{a_i} = \{Cl_1, Cl_2, \cdots, Cl_{m-1}, Cl_m\}$; where $\{Cl_1, Cl_2, \cdots, Cl_{m-1}, Cl_m\}$ are DNF clauses, such that $Cl_j = \{cond_1, cond_2, \ldots, cond_{n-1}, cond_n\}$, $\forall j = \{1, \ldots, m\}$, where $cond_l$ is an atomic condition, $\forall l = \{1, \ldots, n\}$.

Example 17. Let us continue the example depicted in Figure 6.1, assuming that for the flight booking activity, i.e., activity a_1, the requesting user has the following requirements: seat location is “aisle", the class is “economy", the meal is “standard"; or seat location is “windows", the class is “premium economy", the meal is “standard". The requirements are modeled as $UR_{flight} = \{Cl_1, Cl_2\}$, where $Cl_1 = \{[seat, =, \text{“}aisle\text{”}], [class, =, \text{“}economy\text{”}], [meal, =, \text{“}normal\text{”}]\}$ and $Cl_2 = \{[seat, =, \text{“}windows\text{”}], [class, =, \text{“}premiumeconomy\text{”}], [meal, =, \text{“}standard\text{”}]\}$

As introduced in Section 6.1, the privacy-preserving evaluation of user requirements is carried out exploiting the protocol described in [16]. This protocol exploits asymmetric encryption to privately evaluate combinations of predicates in the form *{variable, operator, threshold}*, where the set of supported operators includes: $<$, $>$, $\leq$, $\geq$, $=$, $\neq$. By means of such protocol, it is possible to evaluate whether an encrypted value of a given property satisfies a condition. As an example, it is possible to verify whether the encryption of the price property satisfies the condition {price, $<$,1000}. The key idea is to generate a token ATK for each atomic condition. This is done by using the $GenToken(SK, < cond >)$ function defined in [16], where SK is the user private key generated according to [16]. In

[3]The ontology can defined based on UDDI registers as well as dictionaries used in web service semantic annotation approaches.

order to evaluate whether the encryption of a value v with the corresponding public key PK, i.e., $Enc_{PK}(v)$, satisfies *cond*, we make use of the $Query()$ function defined in [16]. This takes as input the encryption $Enc_{PK}(v)$ and the token ATK generated for *cond* and returns a predefined message M, if *cond* is satisfied, $\perp$, otherwise.

However, to make a service able to perform this protocol, it has to know the name of the property to which the condition applies. This information is included in the atomic condition token, as the following definition states.

Definition 14. *Atomic Condition Token.* Let *cond* be an atomic condition. Let (PK, SK) be a pair of public and secret keys generated by u according to [16]. The Atomic Condition Token for *cond*, $ATK(cond)$, is defined as:

$$(Enc_{PK}(cond.prop.name), GenToken(SK, < cond >))$$

where $Enc_{PK}()$ and $GenToken()$ are defined based on [16], $cond.prop.name$ is the property name included in *cond*.

Example 18. Let us consider the atomic condition for the flight service in Example 16, that is, $cond = [price, <, 50]$. We have $ATK(cond) = (Enc_{PK}(\text{“}price\text{”}), GenToken(SK, sCond))$ where *sCond* is a bit string encoding the threshold and the operator in the predicate *cond*.

By exploiting the above definition, EPApp creates, for each activity a_i in the workflow, a unique data structure, called user requirement tokenset, encoding the user requirements for a_i.

Definition 15. *User requirement tokenset.* Let UR_{a_i} be the requirements of a user u for activity a_i in a workflow WF. Let (PK, SK) be a pair of public secret keys generated by u according to [16]. The user requirement tokenset associated with UR_{a_i}, i.e., $TK_{UR_{a_i}}$, is defined as:

$$\{TKC_1, TKC_2, \cdots, TKC_m\}$$

where $TKC_j = \{\, ATK(\overline{cond}), \forall \overline{cond}$ in the j-th DNF clause of $UR_{a_i}\}$, $\forall j = \{1, \ldots, m\}$.

Once all tokensets have been generated, EPApp searches a web service able to execute the first activity, i.e., a_1, so as to start the composition. Before invoking this service, EPApp has to privacy-preserving evaluate whether the properties of the selected web service satisfy the user requirements on a_1. Since the process for the privacy-preserving evaluation is the same for each activity, in the following we explain it for the generic activity a_{j+1}. In general, once a web service WS_j has completed the execution of the assigned activity a_j, it has to search for a service able to perform activity a_{j+1}. Let assume that WS_{j+1} is the available service

Table 6.1 User requirement enforcement protocol

1.	$WS_j \rightarrow WS_{j+1}$	$(PK, TK_{UR_{a_{j+1}}})$
2.	WS_{j+1}	$SEP = GenerateSEP(PK,$ $Properties(WS_{j+1}))$
3.	WS_{j+1}	$arRes = PrivateEvaluation(SEP,$ $TK_{UR_{a_{j+1}}})$
4.	$WS_{j+1} \rightarrow WS_j$	$arRes, SEP$
5.	WS_j	$Proof(TK_{UR_{a_{j+1}}}, SEP)$

found by WS_j. In order to verify whether WS_{j+1} satisfies the user requirements applied on activity a_{j+1}, that is, $UR_{a_{j+1}}$, WS_{j+1} and WS_j execute the protocol in Table 6.1.

In particular, as a first step, WS_j forwards the public key PK and the user requirement tokenset $TK_{UR_{a_{j+1}}}$ to WS_{j+1}. Then, WS_{j+1} uses the received public key PK to encrypt each of its property names and values. More precisely, assuming that $Properties(WS_{j+1})$ returns all pairs of property names and values associated with WS_{j+1}, the $GenerateSEP()$ function returns the Set of Encrypted Properties (SEP), that is, the set containing, for each $prop \in Properties(WS_{j+1})$, a distinct element. Each element consists of two attributes, where the first one stores the encryption of a property name, i.e., $Enc_{PK}(prop.name)$, whereas the second one contains the encryption of the corresponding property value, i.e., $Enc_{PK}(prop.val)$. In particular, the first attribute of the SEP elements is used to retrieve from the user requirements tokenset $TK_{UR_{a_{j+1}}}$ those atomic conditions (i.e., the corresponding token) that apply to a property owned by WS_{j+1}. Once retrieved, the corresponding encrypted value (i.e., the second attribute in the SEP element) can be passed to the secure evaluation function (i.e., $Query()$), to determine whether the atomic condition is satisfied. This process is executed by $PrivateEvaluation()$, whose pseudocode is given in Algorithm 4.

This algorithm receives as input the set of encrypted properties SEP and the user requirement tokenset $TK_{UR_{a_{i+1}}}$. It returns a data structure, called $arRes$, representing the array of results of the DNF clauses evaluation in the received tokenset. In particular, the j-th element of $arRes$ refers to the j-th DNF clause of the user requirements (i.e., the j-th TKC in $TK_{UR_{a_{i+1}}}$). Every element in $arRes$ is, in turn, defined as an array whose elements store the results of the evaluation of the atomic conditions in the corresponding clauses. As an example, the j-th element in $arRes$ is an array containing an element for each atomic condition (i.e., for each ATK in TCK_j) contained in the j-th DNF clause of $UR_{a_{i+1}}$. In order to generate $arRes$, $PrivateEvaluation()$ iteratively considers each TKC in $TK_{UR_{a_{i+1}}}$, and each atomic condition token ATK in the considered TKC (cfr. the for cycles in lines 3 and 5). Then, it checks whether ATK poses a condition on one of WS_{j+1}'s properties (line 7).

Function 4 *PrivateEvaluation()*

Input: $TK_{UR_{a_{i+1}}}$, SEP
Output: $arRes$;

1: Let initialize $arRes$ with all values set as false;
2: $\alpha = 0$;
3: **for** $\overline{TKC} \in TK_{UR_{a_{i+1}}}$ **do**
4: $\beta = 0$;
5: **for** $\overline{ATK} \in \overline{TKC}$ **do**
6: **for** $el \in SEP$ **do**
7: **if** $(el.Enc_{PK}(prop.name) == \overline{ATK}.Enc_{PK}(cond.prop.name))$ **then**
8: $arRes[\alpha][\beta] = Query(el.Enc_{PK}(prop.val), \overline{ATK}.GenToken(SK, < cond >))$;
9: $break$;
10: **end if**
11: **end for**
12: $\beta = \beta + 1$;
13: **end for**
14: $\alpha = \alpha + 1$;
15: **end for**
16: *return arRes*;

If this is the case, it evaluates the condition using the $Query()$ function defined in [16] and stores the result (i.e., true or false) in the corresponding element of $arRes$ (line 8). Once all TKC have been evaluated, the resulting $arRes$ is returned.

As a final step of the protocol in Table 6.1, WS_{j+1} sends WS_j both $arRes$ and SEP. The first is used by WS_j to verify whether there exists at least a DNF clause whose conditions are all satisfied. In contrast, SEP represents the proof of the correct evaluation of user requirements, in that WS_j can re-execute the same private evaluation as the one performed by $PrivateEvaluation()$ and check the obtained $arRes$. This proof is performed by $Proof()$ using the same logic as $PrivateEvaluation()$. In order to make EApp able to perform this check as well, WS_j sends to it SEP and $arRes$.prop.val

Example 19. Let us consider UR_{flight} introduced in Example 17. EPApp generates a user requirement tokenset $TK_{UR_{flight}}$, including two token clauses, that is, TKC_1 and TKC_2 for Cl_1 and Cl_2, respectively. For each of the token clauses, three atomic condition tokens are generated in turn, that is, $TKC_1 = \{ATK(cond_{1_1}), ATK(cond_{1_2}), ATK(cond_{1_3})\}$ and $TKC_2 = \{ATK(cond_{2_1}), ATK(cond_{2_2}), ATK(cond_{2_3})\}$. Then, EPApp sends $TK_{UR_{flight}}$ and the public key PK to WS_1. Assume that WS_1 has the following properties: $\{(prop =$ "*seat*", $prop.val =$ "*window*"), $(prop =$ "*class*", $prop.val =$ "*premium economy*"), $(prop =$ "*meal*", $prop.val =$ "*standard*"), $(prop =$ "*number of transits*", $prop.val = 1)\}$ from which

a set SEP is generated with the received PK, that is, $SEP = \{el_1 = (Enc_{PK}(\text{“}seat\text{”}), Enc_{PK}(\text{“}windows\text{”})), el_2 = (Enc_{PK}(\text{“}class\text{”}), Enc_{PK}(\text{“}premium\ \ economy\text{”})), el_3 = (Enc_{PK}(\text{“}meal\text{”}), Enc_{PK}(\text{“}standard\text{”})), el_4 = (Enc_{PK}(\text{“}number\ of\ transits\text{”}), Enc_{PK}(1))\}$. WS_1 initializes *arRes* with values “false", the number of elements of *arRes* is equal to the number of elements of $TK_{UR_{flight}}$. After that, it processes the received $TK_{UR_{flight}}$ and SEP, to compare the encryption of its property names with the received encryption of the atomic condition names. With the first token clause TKC_1, WS_1 verifies that the atomic condition name encryptions matches its property name encryptions, however, $Query()$ verifies that not all the atomic conditions are met. Hence, the three first elements of *arRes* are filled with values “false". With the second token clause TKC_2, all atomic condition name encryptions are matched as well, and function $Query()$ verifies that the atomic conditions are met. So, the three next elements in *arRes* are set to “true". WS_1 sends *arRes* and SEP back to EPApp.

6.4 Privacy-preserving provider requirements

As illustrated in Section 6.2, the proposed framework makes service providers able to pose conditions on web services they have to invoke to proceed in the composition deployment. Similarly to user requirements, these constraints are modeled as the disjunctive normal form of a boolean expression over atomic conditions.

Definition 16. *Service provider requirements.* Let a_i be an activity in a workflow WF, modeling the composite web service required by a requesting user u. The requirements specified by the provider associated with activity a_i, referred to as WR_{a_i}, are defined as a Boolean expression over atomic conditions, denoted in the DNF as: $PR_{a_i} = \{PCl_1, PCl_2, \cdots, PCl_{m-1}, PCl_m\}$, where $\{PCl_1, PCl_2, \cdots, PCl_{m-1}, PCl_m\}$ are DNF clauses, such that $PCl_j = \{cond_1, cond_2, \ldots, cond_{n-1}, cond_n\}$, $\forall j = \{1, \ldots, m\}$, where $cond_l$ is an atomic condition, $\forall l = \{1, \ldots, n\}$.

Example 20. Let us continue with Example 17. Let us assume that the web service carrying on the activity *flight* requires that the next service to be invoked must ensure at least 15-second response time, usage of 2-GB RAM, and adoption of AES, or at least 10-second response time, usage of 2-GB RAM, and use of RSA. These provider requirements are modelled as follows: *flight* $PR_{flight} = \{\{[Response\_time, \leq, 15], [RAM, \geq, 2], [Enc\_alg, =, \text{“}AES\text{”}]\}, \{[Response\_time, \leq, 15], [RAM, \geq, 2], [Enc\_alg, =, \text{“}RSA\text{”}]\}\}$.

According to Definition 16, provider requirements are satisfied if there exists at least a clause whose conditions are all verified by the property values of the candidate service, say WS_{j+1}. This means that WS_j and WS_{j+1} have to cooperate to privately verify whether WS_{j+1} property values satisfy the conditions of at least a clause. In general, given a condition

cond applying on a property *prop*, if WS_{j+1} satisfies it it means that WS_{j+1}'s *prop* value(s) belongs to the predefined set of values satisfying *cond*. This set is computed as follows.

Definition 17. *SVS(cond)* Let $cond = [prop, op, thresh]$ be an atomic condition, the set of values satisfying *cond* is defined as:

$$SVS(cond) = \begin{cases} (thresh, UDom(prop)] & \text{if } op = '>' \\ [thresh, UDom(prop)] & \text{if } op = '\geq' \\ [LDom(prop), thresh) & \text{if } op = '<' \\ [LDom(prop), thresh] & \text{if } op = '\leq' \\ thresh \cup R & \text{if } op = '=' \\ [LDom(prop) & \\ UDom(prop)] \backslash thresh & \text{if } op = '\neq' \end{cases} \tag{6.1}$$

where $Dom(prop)$ is the domain of *prop*, $UDom(prop)$, $LDom(prop)$) the corresponding upper and lower bound, and R is a set of random values such that $R \cap Dom(prop) = \emptyset$.

Example 21. Let us continue with Example 20 and consider three atomic conditions: $cond_1 = (Response\_time, \leq, 15)$, $cond_2 = (RAM, \geq, 2)$, and $cond_3 = (Enc\_alg, =, \text{“}AES\text{”})$. The SVS for these atomic conditions are generated, as follows: $SVS(cond_1) = [1, 15]$, $SVS(cond_2) = [2, 500]$, $SVS(cond_3) = [1, 5]$.[4]

In order to verify whether WS_{j+1}'s *prop* value(s), denoted as $WS_{j+1}.prop.val$, belongs to $SVS(cond)$, we make use of the protocol presented in [45], which allows to compute the common elements between two private sets. Applying this protocol for the private evaluation of *cond* implies that WS_j generates a polynomial $P_{cond}(y) = \Sigma_{u=0}^{|SVS(cond)|}(\alpha_u \times y^u)$ such that values in $SVS(cond)$ are all and the only roots of $P_{cond}(y)$. Then, WS_j has to encrypt the obtained $|SVS(cond)|$ coefficients, i.e., $\{\alpha_0, \alpha_1, \ldots, \alpha_{|SVS(cond)|-1}, \alpha_{|SVS(cond)|}\}$ with a homomorphic encryption algorithm.

The encrypted values, denoted $EncCoef$, are then sent to WS_{j+1} together with public key and related secure parameters. In order to check if WS_{j+1}'s *prop* value(s) belongs to $SVS(cond)$, WS_{j+1} encrypts these value(s) and evaluates the polynomial $P_{cond}(y)$, by using the received coefficients. Since all coefficients are encrypted, it actually evaluates $Enc(P_{cond}(y)) = Enc(\Sigma_{u=0}^{|SVS(cond)|}(\alpha_u \times y^u))$, where $\forall y \in WS_{j+1}.prop.val$. Based on [45], given a value $y \in WS_{j+1}.prop.val$, by returning the encrypted value $Enc(r * P_{cond}(y) + y)$, where r is random value selected by WS_{j+1}, WS_j is able to decrypt it but it cannot read the value of y if this element is not in the common set. Indeed, in case $y \in SVS(cond)$, then $P_{cond(y)} \neq 0$, which implies that $Dec(Enc(r * P_{cond}(y) + y))$ returns a randomized value thanks to the presence of r.

[4] In the example, we assumed that the *RAM* property ha [1,500] as domain and that the web service supports five encryption algorithms, each of which is associated with a number in the range [1,5].

Based on the above description, the overall process implies that, as a first step, WS_j initializes the needed security parameters. This is performed by the $SetUp()$ function described by Algorithm 5, which takes as input the provider requirements PR_j and generates the encrypted coefficients for all polynomials $P_{\overline{cond}}(y) = \Sigma_{u=0}^{|SVS(\overline{cond})|}(\alpha_u \times y^u)$, $\forall \overline{cond} \in \overline{Cl}, \forall \overline{Cl} \in PR_j$. Then, given a clause $\overline{Cl}$ in PR_j, $\forall \overline{cond} \in \overline{Cl}$, WS_j and WS_{j+1} execute the above-described process whose underling protocol is depicted in Table 6.2.

Function 5 *SetUp()*

Input: $PR_j = \{ Cl_1, Cl_2, \ldots, Cl_n \}$
Output: $SK, PK, EncCoef$

1: $(SK, PK) = generateSecretParamenters()$;
2: $\alpha = 0$;
3: **for** $\overline{Cl} \in PR_j$ **do**
4: $\quad \beta = 0$;
5: $\quad$ **for** $\overline{cond} \in \overline{Cl}$ **do**
6: $\quad\quad SVS = computeSVS(\overline{cond})$;
7: $\quad\quad PolyCoef = generatePolynomial(SVS)$;
8: $\quad\quad EncCoef[\alpha][\beta] = Encrypt(PolyCoef, PK)$;
9: $\quad\quad \beta = \beta + 1$;
10: $\quad$ **end for**
11: $\quad \alpha = \alpha + 1$;
12: **end for**
13: *return* $(SK, PK, EncCoef)$;

Table 6.2 Private evaluation of $\overline{cond}$ in clause $\overline{Cl}$

1.	$WS_j \rightarrow WS_{j+1}$	$EncCoef[l][m]$, where l (i.e., m) is the position of $\overline{Cl}$ ($\overline{cond}$) in PR_j (i.e., $\overline{Cl}$)
2.	WS_{j+1}	generate $P_{cond}()$ using $EncCoef[l][m]$
3.	WS_{j+1}	$EncValues = Enc(r * P_{\overline{cond}}(v) + v)$, $\forall v \in WS_{j+1}.prop.val$
4.	$WS_{j+1} \rightarrow WS_j$	$EncValues$

6.5 Experiments

In order to demonstrate the efficiency of our proposal, we carried out several experiments to measure the overhead, in terms of time and space implied by the privacy-preserving evaluation of user and provider requirements. All experiments have been run on a PC Intel Duo Core CPU 3GHz, 4GB RAM, and 64-bit Windows 7 Professional OS. Each type of experiment has been carried out 20 times, showing then the average overhead.

6.5.1 Time overhead

This overhead has been estimated by measuring the extra time needed for the generation and processing of the security related information (tokensets and polynomial coefficients). This has been done for both user and provider requirements. Moreover, in order to show how the proposal scales, we have considered different scenarios. In particular, as the key size greatly impacts the overhead of the encryption schemes, we run different experiments by varying the size of the adopted keys. Furthermore, we also measured how the complexity of requirements impacts the overhead. More precisely, we have assumed that user and provider requirements consist of a single DNF clause, but we varied the number of atomic conditions.

Time overhead for user requirement evaluation We implemented the privacy-preserving user's requirement evaluation by using the pairing-based cryptography over elliptic curve presented in [83], with 224, 256, 384, 521, and 603 bits as key size.[5] We also varied the numbers of atomic conditions from 10 to 100. Figure 6.2-a shows the time needed for the generation of the tokenset by EPApp. In the case of the largest key size (i.e., 603 bits) and the highest number of conditions (i.e., 100 conditions), the time overhead is about 13s. In the case of the smallest key size (i.e., 224 bits) and the lowest number of conditions (i.e., 10 conditions), the overhead is around 0.45s. Figure 6.2-b shows the time needed by the service to privately evaluate the user requirements, where with the largest key size and the highest number of conditions, the overhead is about 33.2s. Time is greatly reduced, i.e., about 1s, in case of the simplest scenario (i.e., with the smallest key size and the lowest number of conditions).

Time overhead for provider requirements evaluation

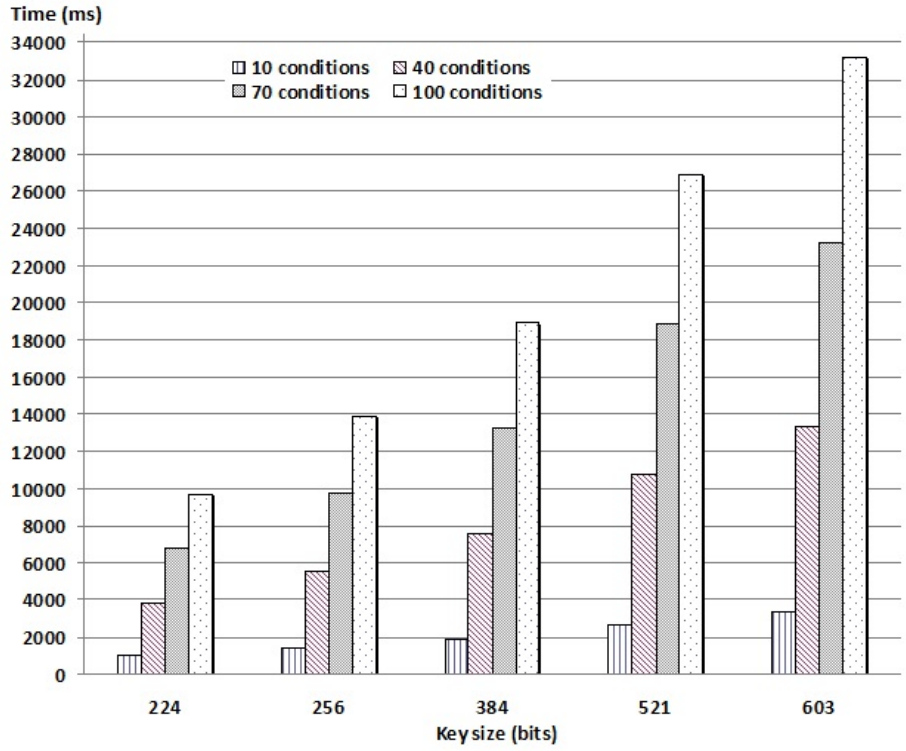
In this experiment we assumed to have WS_j and WS_{j+1} performing the provider requirement evaluation, as described in Section 6.4. In particular, we implemented the framework presented in [45] by using the Paillier homomorphic encryption [112]. Similarly to the previous experiment, we have considered different settings. Here, in addition to the key size and number of atomic conditions, we have to consider that also the degree of the polynomial might impact the performance. We recall that this latter depends on the cardinality of the set of values satisfying the condition (i.e., $SVS(cond)$). As such, we estimated the time by varying all these three factors. In particular, based on NIST recommendation, the selected key size varies in {1024, 2048, 3072, 4096} bits. The polynomial degrees vary from 30 to 100, whereas the number of conditions has been set to 40, 70 and 100. As Figure 6.3 reported, in the worst case (i.e., with 4096-bit key size, 100 conditions and a polynomial of degree 100), at WS_j side the time is about 350s. In the case of the shortest key size (1024 bits), a polynomial degree of 30 and 40 conditions, at WS_j the time is 3s. In contrast, as depicted

[5] These values have been selected based on NIST recommendation.

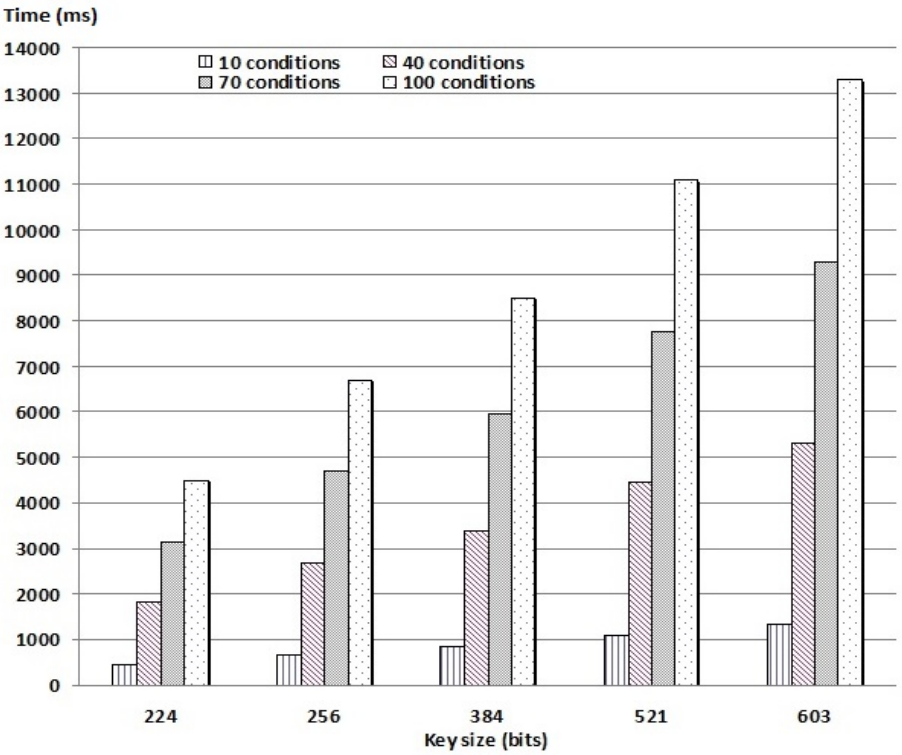
in Figure 6.4, in the worst case at WS_{j+1} side the time is about 4099 milliseconds. In the case of the shortest key size (i.e., 1024 bits), a polynomial degree of 30 and 40 conditions at WS_{j+1} the time is 73 milliseconds. As Figures 6.2 and 6.4 show, we obtained a reasonable time overhead, which we expect can be greatly reduced with more powerful machines, as the ones adopted in the typical SOA architectures.

6.5.2 Space overhead

This overhead has been estimated by measuring the extra information exchanged due to the private evaluation of user and provider requirements. We estimated the SOAP messages size, by varying the number of atomic conditions as well as the size of adopted keys. In particular, the number of atomic conditions varies from 10 to 100. Figure 6.5-a shows the size of SOAP messages generated during the user requirements evaluation. Here, key sizes values are {224, 256, 384, 521, 603} bits. As the figure depicts, in the case of the largest key size (i.e., 603 bits) and the highest number of conditions (i.e., 100 conditions), the SOAP message storing user requirements has size 438KB. In contrast, in the simplest case (i.e., 224 bits key and 10 conditions), the SOAP message size is 17KB. Figure 6.5-b shows the size of the SOAP messages generated during provider requirements evaluation, where adopted key sizes are {1024, 2048, 3072, 4096} bits. In the case of the largest key size and the highest number of conditions, the SOAP message containing the provider requirements has size 123.7KB. In contrast, in the simplest case, the SOAP message size is 13.5KB. We believe this is a reasonable overhead thanks to the actual available bandwidth facilities.

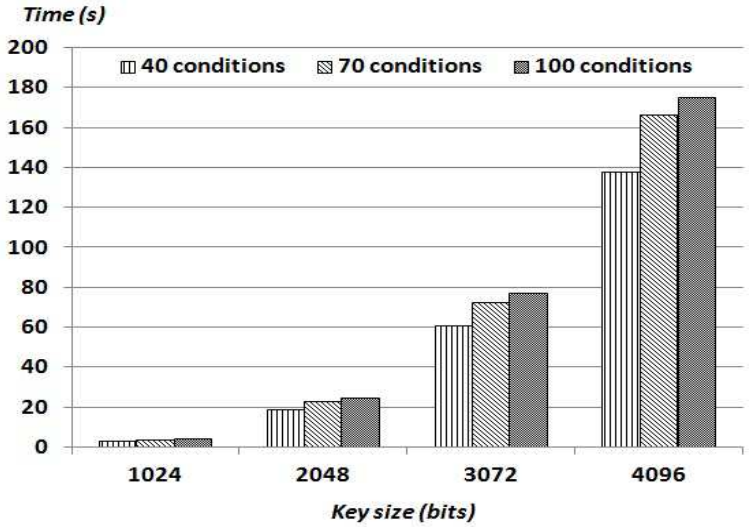


a) Time (ms) for the token generation at the user side

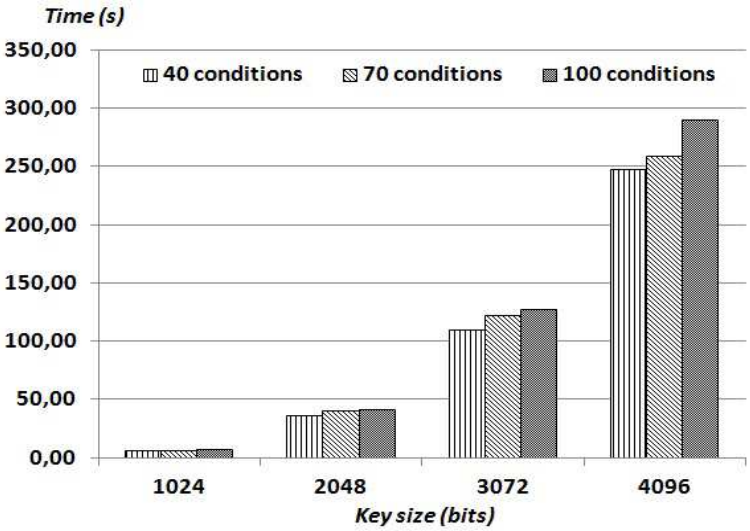


b) Time (ms) for the token evaluation at the service side

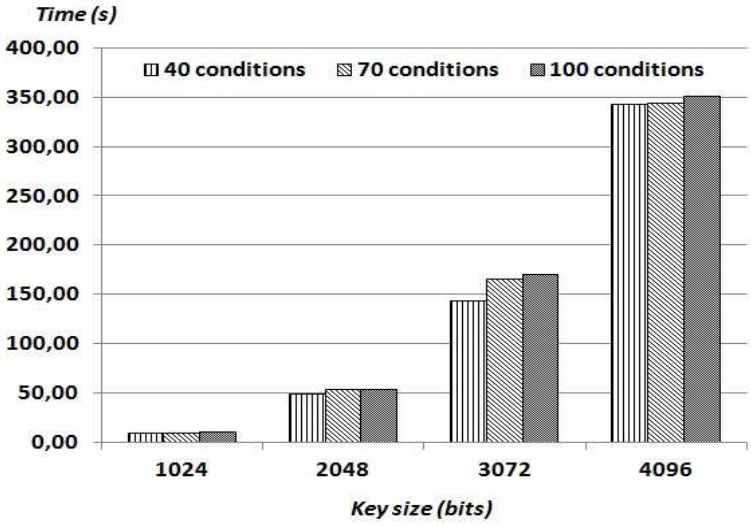
Fig. 6.2 Time overhead for the privacy-preserving evaluation of user requirements



a) with 30 as polynomial degree

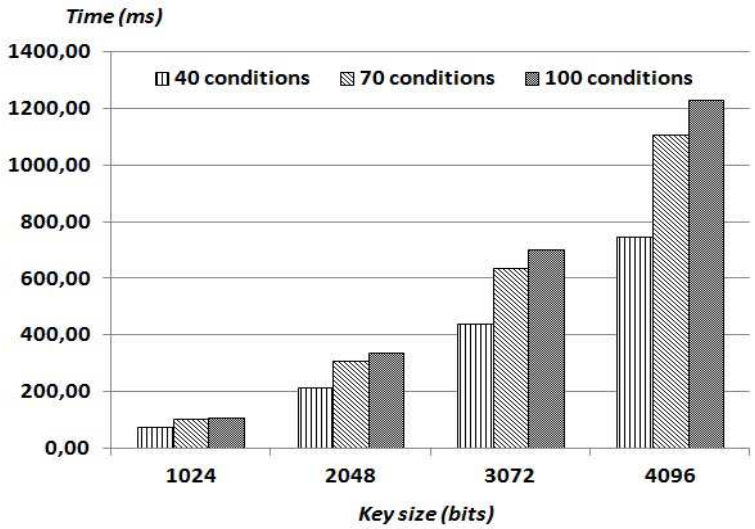


b) with 70 as polynomial degree

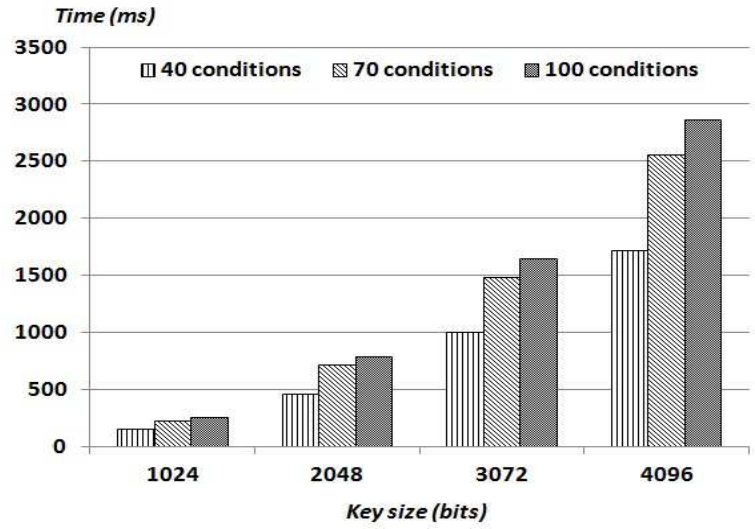


c) with 100 as polynomial degree

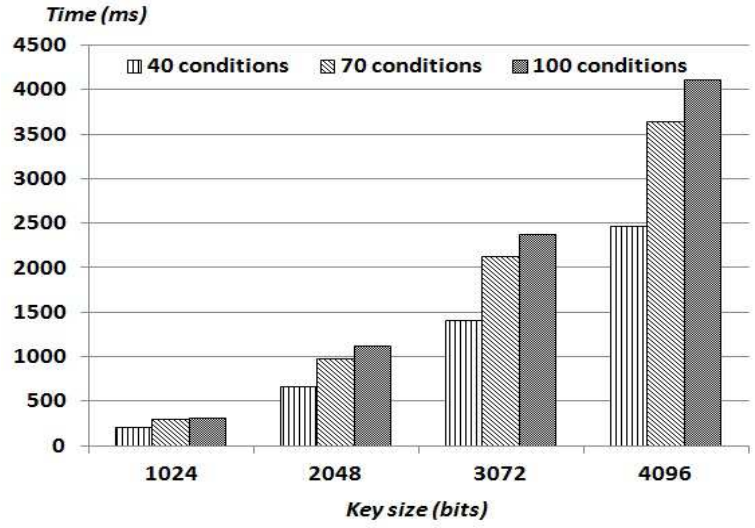
Fig. 6.3 Time overhead for the privacy-preserving evaluation of provider's requirements at WS_j side.



a) with 30 as polynomial degree

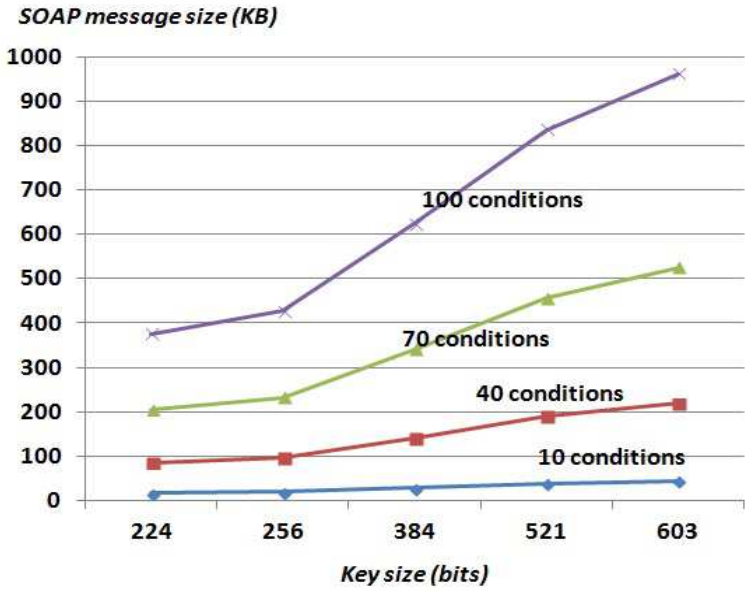


b) with 70 as polynomial degree

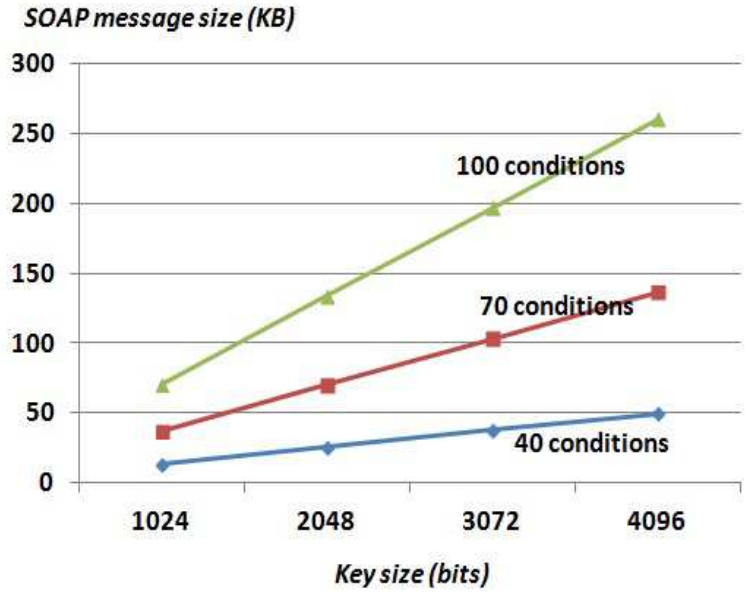


c) with 100 as polynomial degree

Fig. 6.4 Time overhead for the privacy-preserving evaluation of provider's requirements at WS_{j+1} side.



a) Size of SOAP messages created during user requirements evaluation



b) Size of SOAP messages created during provider requirements evaluation

Fig. 6.5 Size of SOAP messages

Chapter 7

Conclusion and Future work

Under the motivation presented in Chapter 1, in this dissertation, we investigate requirements on preserving user privacy and securing user data against adversaries in different collaboration types among multiple participants. In particular, we focus on centralized and decentralized models, and consider participants as users or services. To make it more understanding, we deploy several different collaboration scenarios. Moreover, we investigate and adopt different cryptography algorithms as well, and deploy them in different collaborative scenarios as followed:

- Scenario 1 investigates a secure mobile social collaboration among multiple parties through mobile P2P payment (see Chapter 3). Particularly, in Scenario 1 we presented a privacy-preserving path discovery protocol on support of trust preference enforcement in decentralized mobile payment systems. The protocol is able to protect information about the relationship type, depth and trust of the discovered paths. As well we described a method of optimizing the message flooding using k-anonymity technique and proved that it can improve the network performance.

- Scenario 2 exploits social network relationships for P2P payments over MANET (see Chapter 4). More specifically, in Scenario 2 we presented a privacy-preserving path discovery protocol, namely ESP, in support of trust preference enforcement in decentralized mobile payment systems exploiting MANET. The ESP protocol is able to protect the relationship information in terms of the type, depth, and trust of the discovered paths. We proposed some optimization strategies to decrease the number of tokensets sent over MANET to improve the network performance. We proved the efficiency of the system by experimental results.

- Scenario 3 investigates the web service composition in the centralized model called web service orchestration (see Chapter 5). In particular, in Scenario 3 we developed a

secure protocol for orchestrated composite web services. The protocol does not rely on a trusted broker and, through selective encryption, ensures confidentiality of the involved parameters as well as the correct execution order.

- Scenario 4 investigates the decentralized collaboration among web services according to the decentralized model called web service choreography (see Chapter 6). For more details, in Scenario 4, we presented our proposed framework to enforce user and provider requirements in the scenario of service choreography in a privacy-preserving way, that is, without the release of any information of users and providers relating to their requirements.

However, there are still open problems that need to be solved in the future work as follows:

- In Scenario 1 and 2, there is a need of tackling the case in which nodes suddenly go offline leading to the disconnected path. The messages are blocked at the nodes connecting with the offline node.
- In Scenario 3, the expressive specification language needs to be enriched so as to support users in fully stating their requirements with more details.
- In Scenario 4, there is a need of tackling the case in which two direct connected nodes can collude each other to fool the protocol.

References

[1] Aalst, W., Hofstede, A., Kiepuszewski, B., and Barros, A. (2003). Workflow Patterns. *In Journal of Distributed and Parallel Databases*, 14(1):5–51.

[2] Abdul-Rahman, A. (2004). A framework for decentralised trust reasoning. Ph.D. dissertation, University College London.

[3] Ahmed, T., Tripathi, A., and Srivastava, A. (2014). Rain4service: An approach towards decentralized web service composition. In *IEEE International Conference on Services Computing, SCC 2014*, pages 267–274, Washington, DC, USA. IEEE Computer Society.

[4] Albreshne, A., Fuhrer, P., and Pasquier-Rocha, J. (2009). Web services orchestration and composition: Case study of web services composition. Department of Informatics Internal Working Paper no 09-03, University of Fribourg, Switzerland.

[5] Ali, S. A., Roop, P., and Warren, I. (2013). Web Service Choreography: Unanimous Handling of Control and Data. In *International Journal of Software and Informatics*, volume 7, pages 309–330.

[6] Androulaki, E., Choi, S. G., Bellovin, S. M., and Malkin, T. (2008). Reputation systems for anonymous networks. In *Privacy Enhancing Technologies, 8th International Symposium, PETS 2008, Leuven, Belgium, July 23-25, 2008, Proceedings*, pages 202–218.

[7] Auvinen, A., Vapa, M., Weber, M., Kotilainen, N., and Vuori, J. (2006). Chedar: Peer-to-peer middleware. In *Proceedings of the 20th International Conference on Parallel and Distributed Processing*, IPDPS'06, pages 234–234, Washington, DC, USA. IEEE Computer Society.

[8] Avanade (2013). Global survey: Is enterprise social collaboration living up to its promise? "http://az370354.vo.msecnd.net/social-enterprise/Avanade

[9] Balakrishnan, P. (2011). Enabling web services security between modules in websphere process server v7.0. http://www.ibm.com/developerworks/websphere/library/techarticles/1103_balakrishnan/1103_balakrishnan.html.

[10] Balamurugan, M., Bhuvana, J., and Pandian, S. C. (2012). Privacy preserved collaborative secure multiparty data mining. *Journal of Computer Science*, 8(6):872–878.

[11] Bennett, S. (2014). The history of social networking through the ages. http://www.adweek.com/socialtimes/social-networking-ages/499633.

[12] Benveniste, A. (2008). Composing web services in an open world: Qos issues. In *Fifth International Conference on the Quantitative Evaluaiton of Systems (QEST 2008), 14-17 September 2008, Saint-Malo, France*, page 121.

[13] Biskup, J., Carminati, B., Ferrari, E., Muller, F., and Wortmann, S. (2007). Towards secure execution orders for compositeweb services. In *Web Services, 2007. ICWS 2007. IEEE International Conference on*, pages 489–496.

[14] Blundo, C., De Cristofaro, E., Galdi, C., and Persiano, G. (2008). Validating orchestration of web services with bpel and aggregate signatures. In *on Web Services, 2008. ECOWS '08. IEEE Sixth European Conference*, pages 205–214.

[15] Bogetoft, P., Christensen, D. L., Damgård, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J. D., Nielsen, J. B., Nielsen, K., Pagter, J., Schwartzbach, M., and Toft, T. (2009). Financial cryptography and data security. pages 325–343. Springer-Verlag.

[16] Boneh, D. and Waters, B. (2007). Conjunctive, subset, and range queries on encrypted data. In *Theory of cryptography*, pages 535–554. Springer.

[17] Bosomworth, D. (2015). Statistics on mobile usage and adoption to inform your mobile marketing strategy. http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/.

[18] Bradford, T. and Keeton, W. R. (2012). New person-to-person payment methods: Have checks met their match? "https://www.kansascityfed.org/publicat/econrev/pdf/12q3Bradford-Keeton.pdf".

[19] Brickley, D. and Miller, L. (2010). Foaf vocabulary specification 0.98. http://xmlns.com/foaf/spec/.

[20] Brucker, A., Malmignati, F., Merabti, M., Shi, Q., and Zhou, B. (2013). A framework for secure service composition. In *Social Computing (SocialCom), 2013 International Conference on*, pages 647–652.

[21] Carminati, B., Ferrari, E., and Tran, N. (2015a). A privacy-preserving framework for constrained choreographed service composition. In *Web Services (ICWS), 2015 IEEE International Conference on*, pages 297–304.

[22] Carminati, B., Ferrari, E., and Tran, N. H. (2013a). Enforcing trust preferences in mobile person-to-person payments. In *Social Computing (SocialCom), 2013 International Conference on*, pages 429–434.

[23] Carminati, B., Ferrari, E., and Tran, N. H. (2013b). Smartpay: A lightweight protocol to enforce trust preferences in mobile person-to-person payments. *Science Journal*, 2(4):170–182.

[24] Carminati, B., Ferrari, E., and Tran, N. H. (2014). Secure web service composition with untrusted broker. In *Web Services (ICWS), 2014 IEEE International Conference on*, pages 137–144.

[25] Carminati, B., Ferrari, E., and Tran, N. H. (2015b). Trustworthy and effective person-to-person payments over multi-hop manets. *Journal of Network and Computer Applications*. (accepted).

[26] Caverlee, J., Liu, L., and Webb, S. (2008). Socialtrust: Tamper-resilient trust establishment in online communities. In *Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '08, pages 104–114, New York, NY, USA. ACM.

[27] Chen, K. and Liu, L. (2009). Privacy-preserving multiparty collaborative mining with geometric data perturbation. *Parallel and Distributed Systems, IEEE Transactions on*, 20(12):1764–1776.

[28] Chen, S., Wang, C., and Liu, D. (2011). Secure multi-party collaboration systems in supply chain management. In *1st International Conference on Logistics, Informatics and Service Science*, volume 3.

[29] Cisco (2012). Security practices for online collaboration and social media. "http://www.cisco.com/c/en/us/solutions/collateral/enterprise/cisco-on-cisco/Collaboration_Security-1.pdf".

[30] Co., H. T. (2014). "centralized operations: Strategies for today and tomorrow higher efficiency, better quality, quicker readiness managed services white paper". https://www.evernote.com/shard/s15/sh/a0b8847e-0cd7-4748-8b4d-396c39558c35/200022bfb620e02d.

[31] Dasgupta, D. and Dasgupta, R. (2009). Social networks using web 2.0. http://www.ibm.com/developerworks/library/ws-socialcollab/ws-socialcollab-pdf.pdf.

[32] Davis, I. and E., V. J. (2013). A vocabulary for describing relationships between people. http://vocab.org/relationship/.

[33] Debortoli, S., Sperner, K., Magerkurth, C., Dobre, D., and Meissner, S. (2012). Project deliverable d2.3 orchestration of distributed lot service interations. http://www.iot-a.eu/public/publicdocuments/documents1.

[34] Devi, S. and Thilagavathy, D. (2013). Neighbor node discovery and trust prediction in manets. *International Journal of Science, Engineering and Technology Research*, 2(1):145–149.

[35] Dimitriou, T. and Michalas, A. (2012). Multi-party trust computation in decentralized environments. In *New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on*, pages 1–5.

[36] Dingledine, R., Mathewson, N., and Syverson, P. (2004). Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21. USENIX Association.

[37] Dini, G., Pelagatti, M., and Savino, I. M. (2008). An algorithm for reconnecting wireless sensor network partitions. In *Wireless Sensor Networks*, pages 253–267. Springer.

[38] Domingo-Ferrer, J. (2007). A public-key protocol for social networks with private relationships. In *Modeling Decisions for Artificial Intelligence, 4th International Conference, MDAI 2007, Kitakyushu, Japan, August 16-18, 2007, Proceedings*, pages 373–379.

[39] Domingo-Ferrer, J., Viejo, A., Sebé, F., and González-Nicolás, Ú. (2008). Privacy homomorphisms for social networks with private relationships. *Computer Networks*, 52(15):3007–3016.

[40] Douceur, J. R. (2002). The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260. Springer-Verlag.

[41] Durstenfeld, R. (1964). Algorithm 235: Random permutation. *Commun. ACM*, 7(7).

[42] Eagle, P. N. J., Galbraith, S. D., and Ong, J. (2011). Point compression for koblitz elliptic curves. *Advances in Mathematics of Communication*, 5(1):1–10.

[43] Feldman, A. J., Zeller, W. P., Freedman, M. J., and Felten, E. W. (2010). Sporc: Group collaboration using untrusted cloud resources. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, pages 1–, Berkeley, CA, USA. USENIX Association.

[44] Ferro, E. and Potorti, F. (2005). Bluetooth and wi-fi wireless protocols: a survey and a comparison. *Wireless Communications, IEEE*, 12(1):12–26.

[45] Freedman, M. J., Nissim, K., and Pinkas, B. (2004). Efficient private matching and set intersection. In *Advances in Cryptology - EUROCRYPT 2004*, volume 3027, pages 1–19. Springer Berlin Heidelberg.

[46] Garriss, S., Kaminsky, M., Freedman, M. J., Karp, B., Mazières, D., and Yu, H. (2006). Re: Reliable email. In *In Proc. NSDI*, pages 297–310.

[47] Gartner (2012). Gartner says worldwide mobile payment transaction value to surpass usd 171.5 billion. http://www.gartner.com/newsroom/id/2028315.

[48] Gartner (2014). Gartner says by 2018, more than 50 percent of users will use a tablet or smartphone first for all online activities. http://www.gartner.com/newsroom/id/2939217.

[49] GEOFF (2013). The worldwide growth of social media. http://wersm.com/the-worldwide-growth-of-social-media/.

[50] Gervais, A., Karame, G. O., Capkun, V., and Capkun, S. (2014). Is bitcoin a decentralized currency? *IEEE Security & Privacy*, 12(3):54–60.

[51] Glaeser, E. L., Laibson, D., and Sacerdote, B. (2002). An economic approach to social capital. *Economic Journal*, 112(482):F437–F458.

[52] Golbeck, J. (2005a). Personalizing Applications through Integration of Inferred Trust Values in Semantic Web-based Social Networks. In *Semantic Network Analysis Workshop at the 4th International Semantic Web Conference*.

[53] Golbeck, J. (2005b). Personalizing Applications through Integration of Inferred Trust Values in Semantic Web-based Social Networks. In *Semantic Network Analysis Workshop at the 4th International Semantic Web Conference*.

[54] Golbeck, J., Parsia, B., and Hendler, J. (2003). Trust networks on the semantic web. *In Cooperative Information Agents VII*, 2782:238–249.

[55] Goldreich, O. (2004). The foundations of cryptography. Cambridge University Press.

[56] Gonçalves, M., dos Santos Moreira, E., and Martimiano, L. (2010). Trust management in opportunistic networks. In *Networks (ICN), 2010 Ninth International Conference on*, pages 209–214.

[57] Goyal, P., Parmar, V., and Rishi, R. (2011). Manet: Vulnerabilities, challenges, attacks, application. *IJCEM International Journal of Computational Engineering & Management*, 11(2011):32–37.

[58] Gundelsweiler, G. (2006). Reputation and trust in mobile social networks. In *Location-Based Services in (collocated with International Conference on System of Systems Engineering, SoSe 2006)*.

[59] Hankerson, D. and Menezes, A. (2011). *Encyclopedia of Cryptography and Security (2nd Ed.)*, chapter NIST Elliptic Curves. searching.

[60] Hasan, O., Bertino, E., and Brunie, L. (2009). Efficient privacy preserving protocols for decentralized computation of reputation. at LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon.

[61] Herrmann, K. and Jaeger, M. A. (2004). Payflux - secure electronic payment in mobile ad hoc networks. In Lopez, J., Qing, S., and Okamoto, E., editors, *ICICS*, volume 3269 of *Lecture Notes in Computer Science*, pages 66–78. Springer.

[62] Hoebeke, J., Moerman, I., Dhoedt, B., and Demeester, P. (2004). An Overview of Mobile Ad Hoc Networks: Applications and Challenges. *Journal of the Communications Network*, 3(3):60–66.

[63] Holmes, R. (2015). 5 trends that will change how you use social media in 2015. http://time.com/3590866/social-media-2015/.

[64] Hwang, S. Y. and Lee, C. H. (2013). Reliable web service selection in choreographed environments. *In Journal of Decision Support System*, 54(3):1463–1476. Elsevier Science Publishers B. V.

[65] IBM (2015). Decentralized orchestration of composite web services. http://researcher.ibm.com/researcher/view_project_subpage.php?id=3863.

[66] Isaac, J. T., Zeadally, S., and Cámara, J. S. (2012). A lightweight secure mobile payment protocol for vehicular ad-hoc networks (vanets). 12(1):97–123. Kluwer Academic Publishers.

[67] Ismail, R., Boyd, C., Josang, A., and Russell, S. (2003). Strong privacy in reputation systems. In *4th International Workshop on Information Security Apps (WISA'03)*.

[68] Ismail, R., Boyd, C., Jøsang, A., and Russell, S. (2004). Private reputation schemes for p2p systems. In Fernández-Medina, E., Castro, J. C. H., and García-Villalba, L. J., editors, *2nd Intl. Workshop on Security in Info. Systems (WOSIS'04)*, pages 196–206. INSTICC Press.

[69] Johnson, D., Menezes, A., and Vanstone, S. (2001). The elliptic curve digital signature algorithm (ecdsa). *International Journal of Information Security*, 1(1):36–63.

[70] Jun-feng, T., Yu-zhen, L., and Peng, Y. (2009). A weighted closeness-based trust combination model. In *Proceedings of the 2Nd International Conference on Interaction Sciences: Information Technology, Culture and Human*, ICIS '09, pages 320–325, New York, NY, USA. ACM.

[71] Katiyar, V., Dutta, K., and Gupta, S. (2010a). Article:a survey on elliptic curve cryptography for pervasive computing environment. *International Journal of Computer Applications*, 11(10):41–46.

[72] Katiyar, V., Dutta, K., and Gupta, S. (2010b). A survey on elliptic curve cryptography for pervasive computing environment. *Int. Journal of Computer Applications*, 11(10).

[73] Kattepur, A., Georgantas, N., and Issarny, V. (2013). Qos composition and analysis in reconfigurable web services choreographies. In *ICWS*, pages 235–242. IEEE Computer Society.

[74] Kerschbaum, F., Biswas, D., and de Hoogh, S. (2009). Performance comparison of secure comparison protocols. In *Database and Expert Systems Application, 2009. DEXA'09. 20th International Workshop on*, pages 133–136. IEEE.

[75] Khadka, R., Sapkota, B., Pires, L., van Sinderen, M., and Jansen, S. (2011). Model-driven development of service compositions for enterprise interoperability. In *Enterprise Interoperability*, volume 76, pages 177–190. Springer Berlin Heidelberg.

[76] Kiltz, E., Leander, G., and Malone-Lee, J. (2005). Secure computation of the mean and related statistics. In *Theory of Cryptography*, pages 283–302. Springer.

[77] Kinateder, M. and Pearson, S. (2003). A privacy-enhanced peer-to-peer reputation system. In Bauknecht, K., Tjoa, A., and Quirchmayr, G., editors, *E-Commerce and Web Technologies*, volume 2738 of *Lecture Notes in Computer Science*, pages 206–215. Springer Berlin Heidelberg.

[78] Kinateder, M., Terdic, R., and Rothermel, K. (2005). Strong pseudonymous communication for peer-to-peer reputation systems. In *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC '05)*, pages 1570–1576, New York, NY, USA. ACM Press.

[79] King, B. (2004). A point compression method for elliptic curves defined over $gf(2^n)$. In *Work. Theory and Practice in Public Key Cryptography*.

[80] King, B. (2009). Mapping an Arbitrary Message to an Elliptic Curve when Defined over $GF(2^n)$. *Int. Journal of Network Security*, 8(2).

[81] Kissner, L. (2006). Privacy-preserving distributed information sharing. Document Thesis at School of Computer Science, Carnegie Mellon University, Pittsburgh.

[82] Kissner, L. and Song, D. (2005). Privacy-preserving set operations. In *IN ADVANCES IN CRYPTOLOGY - CRYPTO 2005, LNCS*, pages 241–257. Springer.

[83] Koblitz, N. and Menezes, A. (2005). Pairing-based cryptography at high security levels. In *Proceedings of Cryptography and Coding 2005, volume 3796 of LNCS*, pages 13–36. Springer-Verlag.

[84] Kuter, U. and Golbeck, J. (2007). Sunny: A new algorithm for trust inference in social networks using probabilistic confidence models. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*, AAAI'07, pages 1377–1382. AAAI Press.

[85] Lakhtaria, K. I. and Lakhtaria, K. I. (2012). *Technological Advancements and Applications in Mobile Ad-Hoc Networks: Research Trends*. IGI Global, Hershey, PA, USA, 1st edition.

[86] Lee, K., Jeon, J., Lee, W., Jeong, S., and Park, S. (2003). Web services: Requirements and possible approaches world wide web consortium (w3c). "www.w3c.or.kr/kr-office/TR/2003/ws-qos". accessed on Nov. 30, 2006.

[87] Lenstra, A. and Verheul, E. (2000). Selecting cryptographic key sizes. *Public Key Cryptography*, 1751:446–465. Springer Berlin Heidelberg.

[88] Li, J., Li, R., and Kato, J. (2008). Future trust management framework for mobile ad hoc networks. *Communications Magazine, IEEE*, 46(4):108–114.

[89] Li, J., Zhang, Z., and Zhang, W. (2010). Mobitrust: Trust management system in mobile social computing. In *10th IEEE International Conference on Computer and Information Technology, CIT 2010, Bradford, West Yorkshire, UK, June 29-July 1, 2010*, pages 954–959. IEEE Computer Society.

[90] Li, R. and Li, J. (2013). Requirements and design for neutral trust management framework in unstructured networks. *The Journal of Supercomputing*, 64(3):702–716.

[91] Li, W., Wen, Q., Su, Q., and Jin, Z. (2012). An efficient and secure mobile payment protocol for restricted connectivity scenarios in vehicular ad hoc network. *Computer Communications*, 35(2):188–195.

[92] Lindell, Y. and Pinkas, B. (2000). Privacy preserving data mining. In *Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '00, pages 36–54. Springer-Verlag.

[93] Liu, G., Wang, Y., and Orgun, M. A. (2011). Trust transitivity in complex social networks. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*.

[94] Liu, Y., Ngu, A. H., and Zeng, L. Z. (2004). Qos computation and policing in dynamic web service selection. In *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &Amp; Posters*, WWW Alt. '04, pages 66–73, New York, NY, USA. ACM.

[95] Machanavajjhala, A., Kifer, D., Gehrke, J., and Venkitasubramaniam, M. (2007). L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1).

[96] Malkhi, D., Nisan, N., Pinkas, B., and Sella, Y. (2004). Fairplay: a secure two-party computation system. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 20–20. USENIX Association.

[97] Mani, M., Nguyen, A. M., and Crespi, N. (2009). What's up 2.0 : P2p spontaneous social networking. In Society, I. C., editor, *IEEE INFOCOM '09 : 28th Conference on Computer Communications, April 19-25, Rio de Janeiro, Brazil*, pages 1–2.

[98] Martinelli, F. and Matteucci, I. (2014). Partial Model Checking for the Verification and Synthesis of Secure Service Compositions. In *Public Key Infrastructures, Services and Applications*, pages 1–11. Springer. rank B, FoR1 0803 Thanks to Aniketos, NESSoS.

[99] Matjaz, B. (2009). A hands-on introduction to bpel. http://www.oracle.com/technology/pub/articles/matjaz_bpel1.html. Oracle.

[100] Mendelson, E. (1997). *Introduction to Mathematical Logic (4th ed)*. London Chapman & Hall, Reading, Massachusetts.

[101] Mezzour, G., Perrig, A., Gligor, V., and Papadimitratos, P. (2009). Privacy-preserving relationship path discovery in social networks. In Garay, J., Miyaji, A., and Otsuka, A., editors, *Cryptology and Network Security*, volume 5888 of *Lecture Notes in Computer Science*, pages 189–208. Springer Berlin Heidelberg.

[102] Michalas, A., Olesh, V. A., Komninos, N., and Prasad, N. R. (2011). Privacy Preserving Trust Establishment Scheme for Mobile Ad-hoc Networks. In *IEEE International Conference on Communications*, pages 752–757.

[103] Microsoft (2014). Work like a network: Accelerating team collaboration with social.

[104] Miers, I., Garman, C., Green, M., and Rubin, A. D. (2013). Zerocoin: Anonymous distributed e-cash from bitcoin. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, pages 397–411, Washington, DC, USA. IEEE Computer Society.

[105] Mohamed, E., El-Etriby, S., and Abdul-kader, H. (2012). Randomness testing of modern encryption techniques in cloud environment. In *Informatics and Systems (INFOS), 2012 8th International Conference on*, pages CC–1–CC–6.

[106] Mohamed, M. F., ElYamany, H. F., and Nassar, H. M. (2013). A Study of an Adaptive Replication Framework for Orchestrated Composite Web Services. In *SpringerPlus*.

[107] Moore, K. (2011). From social networks to collaboration networks: The next evolution of social media for business. http://www.forbes.com/sites/karlmoore/2011/09/15/from-social-networks-to-collaboration-networks-the-next-evolution-of-social-media-for-business/.

[108] Nakamoto, S. (2009). Bitcoin: A peertopeer electronic cash system. http://bitcoin.org/bitcoin.pdf.

[109] Nyre, s. A., Bernsmed, K., Bo, S., and Pedersen, S. (2011). A server-side approach to privacy policy matching. In *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on (ARES'11)*, pages 609–614. IEEE Computer Society.

[110] Oracle (2008). Oracle® fusion middleware developer's guide for oracle soa suite, 11g release 1 (11.1.1), e10224-01. http://download.oracle.com/otndocs/products/soa/e10224.pdf.

[111] Oracle (2011). Fusion middleware securing weblogic web services for oracle weblogic server. "http://docs.oracle.com/cd/E21764_01/web.1111/e13713/toc.htm". Oracle Fusion Middleware Online Documentation Library, 11g Release 1 (11.1.1.5).

[112] Paillier, P. (1999). Public-key Cryptosystems Based on Composite Degree Residuosity Classes. In *Proceedings of the 17th International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'99, pages 223–238, Berlin, Heidelberg. Springer-Verlag.

[113] Panackal, J., Pillai, A., and Krishnachandran, V. (2014). Disclosure risk of individuals: A k-anonymity study on health care data related to indian population. In *Data Science Engineering (ICDSE), 2014 International Conference on*, pages 200–205.

[114] Panda, G., Mitra, A., Prasad, A., Singh, A., and Gour, D. (2010). Applying l-diversity in anonymizing collaborative social network. *International Journal of Computer Science and Information Security (IJCSIS)*, 8(2).

[115] Peterson, R. S., Wong, B., and Sirer, E. G. (2010). Blindfold: a system to "see no evil" in content discovery. In Freedman, M. J. and Krishnamurthy, A., editors, *Proceedings of the 9th international conference on Peer-to-peer systems, IPTPS'10, San Jose, CA, USA, April 27, 2010*, page 1. USENIX.

[116] Pietiläinen, A.-K., Oliver, E., LeBrun, J., Varghese, G., and Diot, C. (2009). Mobiclique: Middleware for mobile social networking. In *Proceedings of the 2Nd ACM Workshop on Online Social Networks*, WOSN '09, pages 49–54, New York, NY, USA. ACM.

[117] Pino, L. and Spanoudakis, G. (2012). Constructing secure service compositions with patterns. In *Eighth IEEE World Congress on Services, SERVICES 2012, Honolulu, HI, USA, June 24-29, 2012*, pages 184–191.

[118] Pino, L., Spanoudakis, G., Fuchs, A., and Gürgens, S. (2014). Discovering secure service compositions. In *CLOSER 2014 - Proceedings of the 4th International Conference on Cloud Computing and Services Science, Barcelona, Spain, April 3-5, 2014.*, pages 242–253.

[119] Prathyusha, K. and Ramakrishna, S. S. (2014). Secure multi-party computation protocols for collaborative data publishing with m-privacy. *International Journal of Computer Science and Information Technologies (IJCSIT)*, 5(5).

[120] Ramkumar, T. B., Kalaikumaran, T., and Karthik, S. (2014). Trust based secure payment scheme for multi-hop wireless networks. *International Journal of Advanced and Innovative Research*, 3(5):173–177.

[121] Ripple (2012). Ripple payment system. https://ripple.com/.

[122] Rivest, R., Adleman, L., and Dertouzos, M. (1978). *On Data Banks and Privacy Homomorphisms*. New York: Academic Press.

[123] Rodriguez, J. (2015). Building an iot platform: Centralized vs. decentralized models. https://www.linkedin.com/pulse/building-iot-platform-centralized-vs-decentralized-models-rodriguez.

[124] Rosen, M. (2008). Bpm and soa: Orchestration or choreography - bptrends. http://www.bptrends.com/publicationfiles/0408COLBPMandSOA-OrchestrationorChoreography 0804Rosen v01 _MR_finaldocpdf.

[125] Ross-Talbot, S. and Fletcher, T. (2009). http://www.w3.org/TR/ws-cdl-10-primer/. World Wide Web Consortium, Working Draft WD-ws-cdl-10-primer-20060619.

[126] S., G. R. (2014). New trends in mobile ad hoc network. *International Journal of Ethics in Engineering and Management Education (IJEEE)*, 1(4).

[127] Sarigöl, E., Riva, O., Stuedi, P., and Alonso, G. (2009). Enabling social networking in ad hoc networks of mobile phones. *Proc. VLDB Endow.*, 2(2):1634–1637.

[128] Satish, N. and Wahidabanu, R. (2012). Towards trustworthy semantic qos based web service description and discovery. *International Journal of Soft Computing*, 7:104–112.

[129] Shinde, S. (2011). "academy technotes - centralized/decentralized collaboration model". https://www.evernote.com/shard/s15/sh/a0b8847e-0cd7-4748-8b4d-396c39558c35/200022bfb620e02d.

[130] Singhal, A., Winograd, T., and Scarfone, K. (2007). Guide to secure web services. "NIST Special Publication 800-95".

[131] Squicciarini, A., Carminati, B., and Karumanchi, S. (2011). A privacy-preserving approach for web service selection and provisioning. In *Web Services (ICWS), 2011 IEEE International Conference on*, pages 33–40.

[132] Squicciarini, A., Carminati, B., and Karumanchi, S. (2013). Privacy aware service selection of composite web services invited paper. In *Collaborative Computing: Networking, Applications and Worksharing (Collaboratecom), 2013 9th International Conference Conference on*, pages 260–268.

[133] Srivastava, A. (2014). 2 billion smartphone users by 2015 : 83% of internet usage from mobile. http://dazeinfo.com/2014/01/23/smartphone-users-growth-mobile-internet-2014-2017/.

[134] Staff, D. T. (2014). The history of social networking. http://www.digitaltrends.com/features/the-history-of-social-networking/.

[135] Statista (2015). Statistics and facts about mobile social networks. http://www.statista.com/topics/2478/mobile-social-networks/.

[136] Sweeney, L. (2002a). K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570.

[137] Sweeney, L. (2002b). K-anonymity: A model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):557–570.

[138] Tbahriti, S. E., Mrissa, M., Medjahed, B., Ghedira, C., Barhamgi, M., and Fayn, J. (2011). Privacy-aware daas services composition. In *Proceedings of the 22Nd International Conference on Database and Expert Systems Applications - Volume Part I*, DEXA'11, pages 202–216. Springer-Verlag.

[139] team, I. A. (2015). "adept: An iot practitioner perspective ibm/samsung's proof of concept draft paper for adept". https://www.evernote.com/shard/s15/sh/a0b8847e-0cd7-4748-8b4d-396c39558c35/200022bfb620e02d.

[140] Tebaa, M., El Hajji, S., and El Ghazi, A. (2012). Homomorphic encryption method applied to cloud computing. In *Network Security and Systems (JNS2), 2012 National Days of*, pages 86–89.

[141] ThoughtInfected (2015). Decentralization and the future world order – part i: The revolution is on. http://thoughtinfection.com/2015/01/24/decentralization-and-the-future-world-order-part-i-the-revolution-is-on/.

[142] Tian, J., Lu, Y., and Yuan, P. (2009). A weighted closeness-based trust combination model. In Sohn, S., Chen, L., Hwang, S., Cho, K., Kawata, S., Um, K., Ko, F. I. S., Kwack, K.-D., Lee, J. H., Kou, G., Nakamura, K., Fong, A. C. M., and Ma, P. C. M., editors, *Int. Conf. Interaction Sciences*, volume 403 of *ACM International Conference Proceeding Series*, pages 320–325. ACM.

[143] Toh, C. K. (2001). *Ad Hoc Mobile Wireless Networks: Protocols and Systems*. Prentice Hall.

[144] Tople, S., Shinde, S., Chen, Z., and Saxena, P. (2013). Autocrypt: Enabling homomorphic computation on servers to protect sensitive web content. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, pages 1297–1310, New York, NY, USA. ACM.

[145] Truby, A. (2011). 15 social media collaboration platforms. http://www.freshminds.net/2011/03/social-media-collaboration-platforms-wiki/.

[146] Velloso, P. B., Laufer, R. P., de Oliveira Cunha, D., Duarte, O. C. M. B., and Pujolle, G. (2010). Trust management in mobile ad hoc networks using a scalable maturity-based model. *IEEE Transactions on Network and Service Management*, 7(3):172–185.

[147] Voss, M., Heinemann, A., and Mühlhäuser, M. (2005). A privacy preserving reputation system for mobile information dissemination networks. In *First International Conference on Security and Privacy for Emerging Areas in Communications Networks (SECURECOMM'05)*, pages 171–181. IEEE.

[148] Vu, L.-H., Hauswirth, M., and Aberer, K. (2005). Qos-based service selection and ranking with trust and reputation management. In *Proceedings of the 2005 Confederated International Conference on On the Move to Meaningful Internet Systems - Volume Part I*, OTM'05, pages 466–483, Berlin, Heidelberg. Springer-Verlag.

[149] Vukojevic, K., Karastoyanova, D., and Leymann, F. (2013). Choreographies and cloud infrastructure for simulation workflows. http://www.iaas.uni-stuttgart.de/institut/mitarbeiter/vukojevic/images/20121203_SimTech _Statusseminar_KarolinaVukojevicp̈df.

[150] Walsh, A. E., editor (2002). *Uddi, Soap, and Wsdl: The Web Services Specification Reference Book*. Prentice Hall Professional Technical Reference.

[151] Wang, Y., Zhang, J., and Vassileva, J. (2014). A super-agent-based framework for reputation management and community formation in decentralized systems. *Computational Intelligence*, 30(4):722–751.

[152] White, S. A. (2004). Process modeling notations and workflow patterns. http://www.bptrends.com/.

[153] Wu, J. (2005). *Peer-to-Peer Overlay Abstractions in MANETs*. Handbook on Theoretical and Algorithmic Aspects of Sensor Ad Hoc Wireless, and Peer-to-Peer Networks.

[154] Xiao, B., Cao, J., Shao, Z., Zhuge, Q., and Sha, E. H.-M. (2007). Analysis and algorithms design for the partition of large-scale adaptive mobile wireless networks. *Computer communications*, 30(8):1899–1912.

[155] Xu, W., Venkatakrishnan, V., Sekar, R., and Ramakrishnan, I. (2006). A framework for building privacy-conscious composite web services. In *Web Services, 2006. ICWS '06. International Conference on*, pages 655–662.

[156] Xue, M., Carminati, B., and Ferrari, E. (2011). P3D - privacy-preserving path discovery in decentralized online social networks. In *Proceedings of the 35th Annual IEEE International Computer Software and Applications Conference, COMPSAC 2011, Munich, Germany, 18-22 July 2011*, pages 48–57.

[157] Yang, B., Zhou, M., and Li, G. (2007). A reputation system with privacy and incentive. In Feng, W. and Gao, F., editors, *SNPD (1)*, pages 333–338. IEEE Computer Society.

[158] Yao, A. C.-C. (1986). How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, SFCS '86, pages 162–167, Washington, DC, USA. IEEE Computer Society.