A Service-oriented Middleware for Privacy Protection in Pervasive Computing

Roberto Speicys Cardoso

N^{O} Ordre .
de la thèse .

THÈSE

présentée

DEVANT L'UNIVERSITÉ PARIS-VI

pour obtenir

le grade de: *DOCTEUR DE L'UNIVERSITÉ PARIS-VI*

Mention: **Informatique**

PAR

Roberto SPEICYS CARDOSO

Équipe d'accueil: **INRIA, Équipe-Projet ARLES**

École Doctorale: **Informatique, Télécommunications et Electronique de Paris**

TITRE DE LA THÈSE:

Intergiciel orienté services pour la protection de la vie privée dans les systèmes d'informatique diffuse

SOUTENUE LE 15 / 06 / 2009 devant la commission d'Examen

COMPOSITION DU JURY

Licia CAPRA - University College London, ENGLAND	Rapporteur
Yves DESWARTE - LAAS - CNRS, FRANCE	Rapporteur
Daniel AUGOT - INRIA, FRANCE	Examinateur
Fabio KON - Universidade de São Paulo - IME, BRÉSIL	Examinateur
Jacques MALENFANT - Université PARIS-VI, FRANCE	Examinateur
Valérie ISSARNY - INRIA, FRANCE	Directrice de Thèse

Every night a world created, complete with furniture – friends made and enemies established; a world complete with braggarts and with cowards, with quiet men, with humble men, with kindly men. Every night relationships that make a world, established; and every morning the world torn down like a circus.

At first the families were timid in the building and tumbling worlds but gradually the technique of building worlds became their technique. Then leaders emerged, then laws were made, then codes came into being. And as the worlds moved westward they were more complete and better furnished, for their builders were more experienced in building them.

The families learned what rights must be observed – the right of privacy in the tent; the right to keep the past black hidden in the heart; the right to talk and to listen; the right to refuse help or to accept, to offer help or to decline it; the right of son to court and daughter to be courted; the right of the hungry to be fed; the rights of the pregnant and the sick to transcend all other rights.

And the families learned, although no one told them, what rights are monstrous and must be destroyed: the right to intrude upon privacy, the right to be noisy while the camp slept, the right of seduction or rape, the right of adultery and theft and murder. These rights were crushed, because the little worlds could not exist for even a night with such rights alive.

– *John Steinbeck,* The Grapes of Wrath

Abstract

The widespread usage of computing devices is eroding the individual's right to privacy. Pervasive computing environments, where users interact with a number of computing devices unconscious of their presence, harm user privacy by generating digital trails of every ordinary user activity. Privacy protection is thus a critical factor for the success of pervasive computing applications.

Service oriented pervasive computing, where resources and applications are modeled as services, offers a compelling implementation of pervasive computing. In service oriented computing, applications can more easily handle the openness, heterogeneity and dynamics typical of pervasive computing environments. Realization of this view requires a service oriented middleware that provides the basic features for provision and consumption of pervasive services: namely, service access, service discovery and service composition. The service oriented middleware is particularly critical for privacy protection in service oriented pervasive computing since privacy vulnerabilities at the middleware layer affect any application using the middleware.

In this thesis, we study how a service oriented middleware affects the privacy of users in pervasive computing environments. Our contribution is a privacy-enhanced middleware that increases privacy protection during service access, discovery and composition. The middleware provides a service access protocol that enables nodes to communicate privately without the problems of public key encryption. The protocol splits messages through multiple paths that resist to attackers controlling a certain number of nodes or networks. It also provides a privacy-enhanced service discovery protocol that uses encoded service descriptions to protect personal information and that reduces the trust requirements of service directories. Since different service descriptions can generate the same encoded data, attackers are unable to identify the original service from the encoded description in service announcements and requests. Finally, the middleware provides a service composition mechanism that allows users to compare the privacy impact of executing service compositions that are functionally equivalent but define different partitions of data among service providers, thus enabling selection of the composition that causes the smallest impact on user privacy. The middleware features are implemented and evaluated in terms of performance and effectiveness.

Our middleware architecture facilitates the development of service oriented pervasive applications that respect the privacy of individuals. Since the middleware handles the

privacy issues introduced by the underlying software platform, applications can focus on application-specific mechanisms for privacy protection. Users that consume services on top of this middleware are also able to more effectively protect their privacy, since they can rely on middleware provided functionalities to better control personal information disclosure.

Contents

List of Figures

List of Tables

Every breath you take
Every move you make
Every bond you break
Every step you take
I'll be watching you

The Police, "Every Breath You Take"

1

Introduction

This thesis discusses a crucial issue in the digital society. With information technologies more present in our daily lives and computers playing a role increasingly important in human interactions, how can individuals benefit from those advances and still keep control over their privacy? As video cameras and sensors become prevalent, computer processors grow powerful, network connection is ubiquitously available and storage capacity increases exponentially, the disturbing vision of a surveillance society where all actions of the citizens are observed, registered and analyzed becomes closer to turn into reality. The computer science research community must urgently answer this question: can information systems be designed so as to avoid this assault on privacy?

This question may seem very direct at first, but it hides a myriad of complex issues. Privacy is not a black-and-white concept and we cannot say, for instance, that an individual inside his house has privacy while another individual walking on the streets does not. Rather, privacy can assume different degrees according to multiple factors. On the one hand, total privacy is not achievable nor desirable: the mere fact of living in society obliges individuals to share some personal information to people connected to them such as members of their family, close friends or coworkers. On the other hand, the total absence of privacy reduces human freedom, negatively affects the way we construct social relations and suppresses the individual's free will. The main issue, then, is not how to

design systems that provide privacy to users but instead: *how can individuals effectively manage their privacy in a society where they interact with information systems as part of their everyday lives?*

Before answering that question, we should consider if this debate is relevant at all. One can argue that these issues are old: humans have been living in society for a long time and privacy has always been a concern in human relations. But do computers cause a specific impact on privacy? If so, why is it different from the impact caused by the popularization of other technologies? We can also consider this debate from a more fundamental point of view: why should we care? Is not privacy a value from the past that has no place in the modern society? Cannot we just adapt ourselves to a world without privacy and "get over it" [1]?

In this chapter we motivate the need for privacy protection, particularly in pervasive computing. In Section 1.1 we analyze the conflict between privacy and technology and argue that computers cause a harmful effect on privacy that cannot be compared to any other technology that came before, and that call for specific privacy protection mechanisms. In Section 1.2 we discuss in particular the privacy risks introduced by pervasive computing and define the specific pervasive environment considered during this thesis. Finally, in Section 1.3, we present the document organization and introduce the chapters that follow.

1.1 Privacy and Technology

Throughout the evolution of human societies, technology and privacy have been playing conflicting roles. When photographic cameras became popular in the end of the 19th century, for instance, some people started to question the right to take pictures, and to claim for laws to restrict "the unauthorized circulation of portraits of private persons" [WB90]. This, at the time new technology, made it possible to take unsolicited pictures and to distribute portraits without the consent of the persons portrayed. People lost control over how, when and to whom they reveal their face and their activities. Because of those changes, society had to modify some of its rules to restore the pre-photography conception of privacy, mainly through the creation of laws that limit the right to take pictures. Even though this issue is almost one hundred years old, the growing number of disputes between celebrities and sensational magazines show that a satisfactory balance is yet to be achieved.

The popularization of airplanes also required society to re-consider the notion of privacy. In the mid 20th century, American property law defined that a land owner had property rights not only over the land, but over an indefinite extension above the land [Les04]. At the same time, the law protected the citizen right to privacy in the comfort of his house ("a man's house is his castle" [WB90]). Once again, a new technology conflicted with the contemporary notion of privacy: how could airplanes fly without invading the privacy of

[1] "Privacy is dead, get over it!" - Scott McNealy, Chairman of Sun Microsystems, Gartner Symposium/ITxpo 2001

citizens? Fortunately, this issue was more easily solved: airplanes were not allowed to fly under a minimum altitude. It is interesting to notice, however, that the combination of airplanes and photography allowed for a new way to invade privacy: aerial photographs of people relaxing *inside* their houses. This is a recurring pattern in the relation between technology and privacy: often, technologies designed for totally unrelated purposes are combined in unforeseen ways to create a new form of privacy invasion.

Today, we are once again confronted with the conflict between technology and privacy. Our present society is described as an intermediary step towards a future Information Society or Knowledge Society [Dru69]. A Knowledge Society is one capable to "identify, produce, process, transform, disseminate and use information to build and apply knowledge for human development" [B+05b]. In knowledge societies, computers and networks play a key role to facilitate, develop and increase the flow of information. As such, we can expect computers and the Internet to become each day more present in our lives. The computer science research community must identify the risks to privacy created by the widespread usage of computers, and propose solutions that restore the equilibrium between privacy and technology to contribute with the creation of a future society where privacy continues to be a fundamental right.

In this thesis we argue that privacy is an essential value in any society and a basic human necessity. We also argue that computers change dramatically the relation individuals have with their personal information. As a result, the future Knowledge Society must consider privacy as one of its core values and take the necessary steps to guarantee this right of every human being [Uni48]. In the particular case of information systems, we believe that architectures play a key role to ensure that individuals have the means to control disclosure of private information and to control how personal data can and cannot be used. It is our understanding that effective privacy protection can only be achieved if systems are based on top of architectures aware of their effects on privacy, designed to make privacy attacks harder to perform and that give users control over private information disclosure.

We present, in Section 1.1.1, a privacy definition that, even though far from being able to capture all the aspects of privacy, serves as a solid base in the context of computer systems for the discussions that are presented in the following chapters. In Section 1.1.2 we argue that privacy is a natural human need and a human right, and as such must not be abdicated. In Section 1.1.3 we claim that computers have a particular harmful effect on individual privacy that cannot be compared to any other technology created before computers, and in Section 1.1.4 we highlight the role of software architectures for privacy protection in computer systems. This discussion of computers and privacy in general introduces the analysis of privacy and pervasive computing in the section that follows.

1.1.1 A Working Definition of Privacy

Privacy is an abstract concept, hence, hard to capture with words. Each individual has a particular view of what is private, and this view is influenced by cultural and generational values. In some tribes, for instance, individuals do not have access to a private area, and it is common to find whole families, including uncles and cousins, living under the same

roof without walls or doors, sharing the same collective space [Wes67]. In other cultures, citizens refuse to carry any form of government-issued identification card and view such cards as a major invasion of privacy[2]. An additional problem is that the term "privacy" is overloaded, and it is currently used in different contexts whenever there is the need to justify the confidentiality of information, from the right to protect trade secrets [CG98], passing by celebrity image rights [Hay05], to the right to abortion [Coh92].

One of the earliest definitions of privacy is the 19th century article from judges Warren and Brandeis, defining privacy as "the right to be let alone" [WB90]. This early definition was created in a time where surveillance required physical presence. To search for documents in possession of a citizen, police had to enter his house and search through his belongings; to spy what the citizen was doing, someone had to follow him; to listen to someone's conversation, police had to be physically close to the people discussing. The right to be let alone, then, was the right of not being spied, of not being victim of unjustified searches.

This definition fails to capture the meaning of privacy in today's society. Indeed, when we are alone in our room navigating the Internet, listening to music inside our houses or reading an e-book sitting on a park bench, we feel that we are alone even though our privacy is being invaded on each of those cases. One can imagine a country – let us call it Surveilland – where every street is under government surveillance, every store, every restaurant and every office. The only places free from the government interference are the homes of Surveilland residents. The population of this country would have the right of being alone inside their houses (the right to privacy according to the above definition by Warren and Brandeis), but this scenario is far from what we would consider an ideal privacy-respectful society.

Others have also tried to provide a definition of privacy better adapted to modern society. However, since there are many aspects of privacy that must be considered, the resulting definitions are too abstract and difficult to use in a concrete scenario. For instance, [IP99] defines privacy as "freedom or immunity from the judgement of others", but what exactly is the judgement of others? When we are buying groceries in a supermarket, do other customers judge us because of the contents of our shopping carts? Should it be considered a privacy invasion? Is there any difference if the shopping cart contents are disclosed to a rewards program? Applying this definition to real world scenarios makes it hard to identify situations where privacy invasions occur and to categorize invasions according to their impact.

Another definition describes privacy as "the interest that individuals have in sustaining a personal space, free from interference by other people and organizations" [Cla99]. Defining privacy as an individual's interest implies that it has to be balanced with other interests of the society. For instance, individuals also have interest in being wealthy, but this interest has to be balanced against other society interests such as guaranteeing access to basic healthcare and residence for all. It is also difficult to determine when other people and organizations are interfering with the individual's personal space. Passive monitoring techniques, such as phone bugs and Digital Rights Management (DRM) software contained

[2] www.no2id.net

in music players and e-book readers, cause little or no interference but can be used for major privacy invasions.

Some authors prefer to define privacy by determining in which situations privacy is lost. The work of [Mar01], for instance, argues that a privacy intrusion happens when a personal border is crossed. Privacy intrusions are categorized in four groups: when a natural border is breached (clothes, walls), a social border is breached (reading over the shoulder), a spatial border is breached (data accessed when it should have been already deleted) and when a protocol border is breached (the expectations of one party are not respected). The categorization of privacy invasions helps us to understand in which situations we feel that we lost privacy and why we have this feeling, but it is still not a definition of privacy.

Attempts to provide a more concrete definition faced the problem of having to convey multiple meanings on a concise form. Some tried to identify the different aspects of privacy and analyze each one independently, for instance dividing privacy into physical privacy, informational privacy and proprietary privacy [All99], or describing what privacy is about and what it is not [Gar00]. When we examine those definitions more closely, it becomes clear that there is a relation between privacy and control. The work presented in [SS73] goes a step further in that direction and defines privacy as "[t]he ability of an individual (or organization) to decide whether, when, and to whom personal (or organizational) information is released". This definition tries to express the notion of individual privacy as well as the idea of corporation privacy, which is more associated to the confidentiality of trade secrets, or government privacy, more related to information confidentiality.

Based on those works, privacy could be defined as *the right to control personal information disclosure*. Even though this is a good working definition since it is clear and concrete, it still falls short in capturing the complexity of privacy. Citizens of our fictional city Surveilland have control over personal data disclosure, but do not have privacy. All-or-none controls are not enough to guarantee privacy; individuals must have multiple mechanisms that enable gradual disclosure of personal information. Moreover, disclosure control can do little for privacy if individuals are not conscious of the risks involved in revealing personal information to third parties. For instance, most people do not know that data collected by loyalty cards can be used as evidence on a court to infer profile and behavior [Gar00, Neb05, Spr04, Sav05] even though identifiable information contained on those cards is never verified and it is fairly easy to create a loyalty card under another person's name. Unaware of those privacy invasion risks, people voluntarily disclose personal data in exchange of 5% discounts. Disclosure control, ignoring invasion risks, is not effective in improving individual privacy.

Thus, the definition of privacy we will be using throughout this thesis is the following:

> *Privacy is the power to accurately control personal information disclosure while being aware of its effects.*

Consequently, whenever individuals do not have the power to control in detail how and when to disclose personal data, they might be victims of a privacy invasion. Similarly, when individuals do not have enough knowledge about the consequences of disclosing personal data, they are unable to perform informed disclosure decisions and can also be victims of

privacy invasions. This definition cannot be used to represent privacy in less conventional contexts where privacy is related to possession such as privacy in trade secrets or private properties. However, in this thesis we are interested in studying the threats to individual privacy posed by computers in the Information Society, and this definition serves as a solid base for the discussions to follow.

Citizens of Surveilland cannot control in detail how to disclose personal data, and thus, according to our definition, they have small or no privacy. Similarly, a consumer that owns a loyalty card but that is not informed on how his data can be used also has small privacy. To increase their privacy, individuals must be able to control data disclosure more granularly and have more information on the impact of revealing personal information.

1.1.2 The Need for Privacy

But do we need privacy at all? Is it necessarily bad that citizens do not have detailed control over disclosure of personal information? Some argue that if one has nothing to hide, there's no need to worry about widely disclosing personal details and that anyone who is not afraid of the past does not need privacy at all.

The main fallacy in those arguments, as already pointed out by others [Sch06, Gar00], is that they assume that privacy is necessarily related with hiding a mistake or a transgression, which is simply not true. We need privacy for the simplest things in life: to read a book, to go to the bathroom, to sing a song while taking a shower, to rehearse a presentation or to play with our kids. Privacy is not secrecy; privacy is related to much more fundamental feelings such as curiosity, independence and ultimately liberty.

The distinction between privacy and hiding secrets can be exemplified by the relation between privacy and intellectual liberty [Coh03b]. To develop their intellect, humans need to have contact with knowledge in multiple forms. We read books on various topics, written using different styles; we listen to all sorts of music, containing explicit lyrics or not; we contemplate paintings from different schools ranging from classical to contemporary. Intellectual growth depends on the individual power to explore diverse forms of culture that express different opinions and points of view. It is natural that, when sharing those experiences with others, individuals will choose what to share according to the characteristics of each social group to which they belong. Someone who reads a book about the horrors of the Holocaust may understandably avoid to discuss it with jewish friends, or a professional classical violinist can decide not to discuss the latest album from Britney Spears with other professional musicians. The fact of being observed while consuming culture reduces the individual's liberty to investigate forms of cultural production that are unpopular among his peers and contributes to create groups with radical views that are shut to external influences.

Privacy is also a key component in determining how humans relate to each other and build their social networks [Str04]. One of the forms that humans use to develop more intimate relationships is by disclosing personal details to people they trust more. It is not a coincidence that individuals have few intimate friends among many others: to build an intimate relationship takes time and effort, and it would be too demanding to become

intimate of all friends and family members. Intimate relations depend on an excluding condition, and privacy gives the means to develop intimacy [Sch77]. Love, friendship and trust can only exist if the persons involved agree on a certain level of privacy [Fri84].

Some organizations in contemporary society are also founded on the concept of privacy. Self-help groups such as the Alcoholics Anonymous[3] or Narcotics Anonymous[4] try to help people to recover from their addiction by putting them in contact with other members that are in the process of recovering or that already recovered from the same problem. Meetings involve members describing personal experiences concerning the group's subject (alcohol or drugs in the above examples) to each other. Such meetings can only succeed if they take place in a private environment and if participants agree to not disclose the discussions to non-members. Privacy in this case is fundamental to allow individuals to reveal their particular experiences without fear of being judged by others.

In spite of the above arguments, some claim that the destruction of privacy would bring more benefits than harm to the society as a whole. One proverbial example is the Panopticon, proposed by the Utilitarian philosopher Jeremy Bentham in 1787 [Ben95]. According to Utilitarianism, every action can be judged as good or bad depending on its impact to society as a whole. Any action that causes the greatest good to the greatest number is considered morally correct. Faithful to these ideas, Bentham proposed the creation of a special type of prison – which he called Panopticon – that would be more effective and require a smaller number of wardens. This prison would be constructed as a circle, with cell's doors towards the center. In the middle of the building would be built a watchtower specially designed to allow wardens to see the prisoners, but not the opposite. Figure 1.1 shows a blueprint of one Panopticon design.

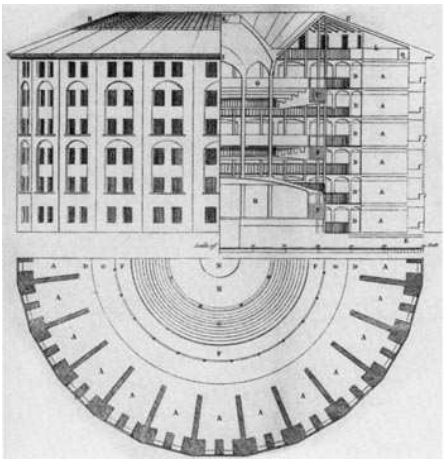


Figure 1.1: Jeremy Bentham's Panopticon

According to Bentham, since prisoners could not see the watch guards, they would feel under surveillance all the time. This sentiment of being constantly watched would inhibit

[3] www.aa.org

[4] www.na.org

convicts to disobey the prison rules, facilitating control. Furthermore, wardens would not need to be on duty all the time because prisoners cannot see them and cannot tell if someone is really watching them or not. This design, thus, allows the greatest number of prisoners to be controlled by the smallest number of guards. The continuous surveillance of prisoners was deemed moral because it caused a greater good to all society, namely, the creation of more efficient and cheapest prisons. Even though Bentham did not succeed to build a Panopticon during his lifetime, it became a popular metaphor to discuss situations where the feeling of being watched can transform the behavior of those under surveillance. For instance, the proliferation of closed-circuit TV (CCTV) cameras in public areas in London [MN03] caused specialists to compare the city to a modern Panopticon[5].

Perhaps the most popular dystopian view of future societies that have no respect for individual privacy is suggested by George Orwell in his novel *1984* [Orw50]. Even though the main subject of the book is politics and the effects of an imaginary authoritarian government over the lives of its citizens, the novel became famous for its description of the government's surveillance mechanisms, personified by its leader Big Brother, to control the population. Telescreens - a sort of two-way television - are present everywhere and continuously turned on, alongside posters containing the inscription "Big Brother is watching you!". The government's Thought Police uses the images from telescreens to identify and to punish citizens capable of merely thinking about challenging the established authority. This omnipresent surveillance mechanism has devastating psychological effects over the main character, and shows how continuous vigilance affects the individual's reasoning and his free will.

Another dystopian vision of a world without privacy is presented by science-fiction writer David Brin. In his book *The Transparent Society* [Bri99], from which an excerpt was published in Wired Magazine [Bri96], Brin argues that there is no turning back: the proliferation of video cameras and surveillance technologies cannot be interrupted. In face of this reality, society has two options: either live with the illusion of privacy while governments and powerful corporations keep secretly using surveillance technologies against citizens and consumers, or accept the end of privacy, install surveillance technologies everywhere and give society the means to have access to that technology.

The main difference between these two options is that while in the first scenario, for instance, only the police would see the images captured by hidden street cameras, in the second scenario everyone knows that every street has a camera and anyone can access its images. In short, Brin claims that the only answer to government monitoring is civilian monitoring. In the future society, according to his view, all citizens will know that they are being watched all the time, at least outside their houses. But they will also be able to watch the others, for instance politicians and the police. The same mechanism that would reduce crime on the streets would also serve to reduce police abuse. Afraid of being watched, all the society members would strictly follow society rules to the benefit of everyone.

Many believe that this is the sole alternative to reclaim the liberty that surveillance technologies have already taken and will continue to take from us. Some researchers dedi-

[5] www.urbaneye.net

cate their time to develop *sousveillance* technologies, or technologies that enable common people to watch and record everything that happens around them, creating the necessary tools to support the "Transparent Society" proposed by Brin. One example is Steve Mann, whose research on wearable computing led to the development of increasingly smaller devices for visually recording all activities of an individual, as shown in Fig. 1.2. Mann goes as far as to argue that such wearable devices actually increase the individual's privacy, since they allow someone to better control which information about himself is disclosed to others [Man01]. According to Mann, computers can mediate all input information used by a human and all output information produced by a human. This mediation enables humans to control with higher precision which personal information they disclose and hence increase their privacy.

Figure 1.2: Evolution of Steve Mann's Inventions

A deeper analysis of the "Transparent Society" argument, however, shows at least two fundamental errors. The first, as pointed out by [Sch08], is that it assumes that all the members of the society have the same power. If that is the case, this argument – that mutual disclosure of private information will give everybody the same power – holds. However, in our present society this is not the case. If a candidate on a job interview reveals his medical records to the interviewer, the company may use that data to refuse him the job. If the interviewer reveals his medical records to the candidate at the same time, the situation does not change. As the interviewer has more power than the candidate on the context they are interacting, disclosure of private data has a greater effect on the candidate than on the interviewer.

The second problem is that this "Transparent Society" is far from what would be considered a free society. It is not a coincidence that the Panopticon was designed primarily as a prison: the idea of being continuously observed changes the behavior of those under observation and reduces their liberty. Even if people are not actively being monitored, the simple hypothesis that their actions *could* be registered or viewed by someone else takes away their freedom to act spontaneously. In such society, people would eventually lock themselves inside their houses afraid of the consequences of going out. Mass usage of surveillance equipment invades the privacy, intimidates, and reduces the individual's free will, regardless of who is controlling the technology.

Privacy is a fundamental requirement to ensure essential human liberties. However, two of the oldest documents defining human rights, the United States Constitution [Gov88] and the *Déclaration des Droits de l'Homme* [Gou89] do not clearly state privacy as one of the rights of man. The first does not contain the word "privacy" in the whole document, while the second states that "The goal of any political organization is to guarantee the natural and imprescriptible human rights. Those rights are: the right to liberty, the right to property, the right to security, and the right to resist to oppression". In spite of being essential to assure each of those rights, privacy is never mentioned as a right by itself.

Only in the more recent Universal Declaration of Human Rights [Uni48], signed by all 191 United Nations member countries, privacy is clearly defined as a right ("No one shall be subjected to arbitrary interference with his privacy, family, home or correspondence, nor to attacks upon his honor and reputation. Everyone has the right to the protection of the law against such interference or attacks."). Other documents based on the Universal Declaration of Human Rights also recognize privacy as a right in different contexts such as the International Covenant on Civil and Political Rights, the International Convention on the Protection of the Rights of All Migrant Workers and Members of Their Families and the Convention on the Rights of the Child [EP06].

In summary, even though challenged by some, privacy is widely recognized as an instinctive human necessity and an important component of the process of human learning and intellectual evolution as well as on the creation and management of social relationships. Despite being overlooked as a right in our early legislation history, the evolution of laws and society allowed for a better understanding of the role of privacy in human organizations and today many documents describe privacy as a human right in different circumstances. Finally, the portrayal of alternative societies and environments where individuals no longer have the right to privacy resulted in dark scenarios that put at stake the individual's free will and liberty. Based on these arguments, we claim that privacy is an important concept and it is in our best interest to protect this right in the future Information Society, where computers will permeate most human interactions.

1.1.3 Impact of Computers on Privacy

Computers changed the way governments and companies process data. Before computers were invented, data collection was error-prone, data processing was onerous and data storage was burdensome. The first punch machines, introduced in 1889, reduced the time required to process the United States census from 18 weeks to only 6 weeks, beginning a long series of improvements in processing time that simplified data handling. Storage remained an issue though, since punch cards were made of paper and their storage required huge areas. In 1943, only seven years after its creation, the american Social Security Board had more than 100 million paper files, that filled six and a half acres of space [Gar00]. This problem lasted until the 60's when magnetic data storage devices were introduced, which reduced the space required to store data and allowed for the collection of more personal information, first by governments and later by companies.

Prominent law professor Alan F. Westin alerted that this trend towards digitalization

was a major threat to the citizen's privacy. According to him,"almost inevitably, transferring information from a manual file to a computer triggers a threat to civil liberties, to privacy, to a man's very humanity because access is so simple" [Wes72]. Recognizing the danger to privacy posed by computers, data protection laws were created at the same pace as computers became more popular. The first such law was introduced in the Land of Hesse, one of the 16 states that constitute Germany, in 1970. After that the Sweden (1973), United States (1974), Germany (1977) and France (1978) also created laws protecting the citizen from data collection [Fla89]. Such laws were necessary because computers change the way data is processed. Those changes can be classified in seven different dimensions: the six dimensions mentioned by [Bri67], namely acquisition, access, dissemination, retention, revision and destruction, plus copy. We detail below the impact of computers on each of those dimensions:

Acquisition: Technologies such as bar codes and RFID sensors allow for data acquisition that is virtually error free if compared to a clerk writing product names on a paper or a human counting the number of persons passing through a door. The consequence is that it became easier to collect data accurately.

Access: Computers enable more people to access the same data simultaneously. With digital data, files can be accessed simultaneously by various users or systems physically far from each other. This possibility enables the creation of multiple applications processing the same set of data, at the same time, for different purposes.

Dissemination: As digital data is virtual, has no weight and no size, it is easier to disseminate it if compared to data kept on paper. It is much simpler, faster and cheaper to send a 100 page document around the world in digital form than when it is printed. As a result, private data captured in one place can be easily transferred to a remote location for correlation and processing.

Retention: Digital data occupies much less space than its equivalent in other formats, and storage media with increasing capacity are constantly being developed. Storage devices with Terabyte capacity can now easily be acquired by any home user. As the storage costs become cheaper, personal data collected and stored in digital form can potentially last forever.

Revision: As digital data does not have a physical representation, it can be easily modified or updated. This reduces the cost of maintaining big databases and encourages the accumulation of as much personal information as possible.

Destruction: An often neglected aspect of digital information is that it is harder to destroy than information stored on other medias (e.g., paper). Burning a hard disc or formatting it before disposal does not always guarantee that its data will also be destroyed. In 2006, Afghan stores were selling used american USB keys found on the garbage of the american military base in Afghanistan, containing secret documents and details about the United States war on terror [Wat06]. This difficulty causes

problems to control availability of digital data. As a consequence, once private information is collected, it can last for longer than intended even when measures have been taken to destroy it.

Copying: Bits can be copied effortlessly without loss of quality as many times as needed. After collection, private data can be easily copied without losing accuracy and individuals can no longer control how many copies of their personal data exist.

The trend towards more storage space, better processing performance, easier distribution and automated data collection continues to increase until today. New technologies such as micro-sensors (also called smart dust [WLLP01]) and tiny video cameras enable the seamless collection of more personal data. Storage space becomes cheaper everyday and data can potentially be inexpensively stored forever. Network evolution has created technologies that allow devices to be almost always connected, using an infrastructure simple to deploy, thus making it easier to transmit data. And finally, the evolution of processors (dual core processors are now commonplace, for instance) and the creation of new data processing algorithms improved the performance of data analysis. Not only computer-related technology is evolving but also it is becoming cheaper, so we can expect that more and more computers will permeate our daily activities in future societies.

There is no other technology that caused an impact so profound on how humans deal with information. Moreover, as computers can be used in numerous applications, most areas of human activities are affected by the use of computers and their potential risks to privacy. Never before a technology as powerful as computers had so much penetration in our daily lives. As a result, the privacy risks caused by computers affect society more extensively than any other technology previously created. We must understand better how humans behave when interacting with computers to design mechanisms that effectively protect privacy in computer systems.

1.1.4 The Role of Architectures

According to Professor Lawrence Lessig, an authority in the law of cyberspace, four constraints regulate human behavior in general: social norms, laws, market and architecture [Les06]. He uses the term "architecture" to describe more generally the way a technology or an object is constructed. It is not restricted to the software architecture or the hardware architecture; it includes all design decisions involved to create the technology. For instance, a home printer can print pages up to a certain resolution: this is an architectural constraint. Due to this architectural constraint, home printers cannot be used to print good-quality fake money. We discuss the meaning of "architecture" as used by Lessig in the context of pervasive computing in Section 1.2.

Lessig gives the example of smoking: a smoker normally asks if he can smoke when in someone else's house (social norms) but it is forbidden to smoke on a plane (law). He may smoke more cigarettes if they become less expensive (market) or if they contain less nicotine (architecture). Those constraints have an effect over the human behavior, but also on each other. For instance, if cigarettes have a weaker smell (architecture), people may

tolerate smoking in restaurants (social norms). Figure 1.3 shows these four constraints and their relation to the human (in bold lines) and to each other (in empty lines).

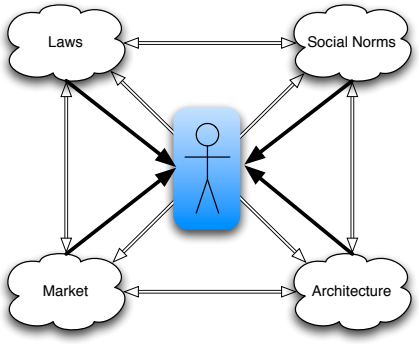


Figure 1.3: The Four Constraints that Regulate Human Behavior

Some may argue that legislation and privacy agreements – the law dimension – suffice to avoid the malicious usage of pervasive applications. The argument is that if, for instance, the government creates a law stating that personal data collected by pervasive systems must not be cached, or if a privacy agreement guarantees that a user's private data will not be historically stored, people do not have to worry about attacks based on the historical analysis of personal data. This, however, is not true. Philip Zimmermann, creator of the Pretty Good Privacy encryption software[6], pointed out in 1996 that even though technology infrastructures tend to last for generations, laws and policies can change overnight. Once an architecture adapted for surveillance is in place, a small transformation in the political conditions may lead to abuse this technology [Zim96]. Whenever a system is designed in a way that favors collection of personal data and privacy is based solely on a law or on an agreement, a small change in law (e.g., after the election of a new president) or in a privacy agreement (e.g., when a company is merged to another) instantly transforms a privacy respectful system into a surveillance tool.

One example of the scenario described by Zimmermann happened in April 2008. Brazilian justice requested data related to the profiles of more than 200 users of Google's social networking website Orkut[7], suspected of promoting pedophilia online [Age08a]. Those profiles contained photo albums protected by a Google-provided privacy-enhancement tool, and those photos could not be accessed by the justice using the website tools. According to Orkut's privacy policy[8]:

> Google provides a number of tools to restrict who can view your profile and other personal information. When you create your profile, look for the "golden key" icon, which allows you to limit viewing certain information to yourself,

[6] www.pgpi.org

[7] www.orkut.com

[8] www.orkut.com/html/en-US/privacy.orkut.html?rev=4

> your friends, friends of friends, or to make the information available to all orkut members.

Fourteen days after the request, Google handed over all data to the brazilian justice [Age08b], even though investigators were not friends nor friends of friends of the users under suspicion. Despite the existence of a privacy agreement that protected the user from privacy invasions, the service architecture allowed for such invasions to occur. All it required was a formal order from the government.

All four constraints mentioned by Lessig – social norms, laws, market and architecture – have an impact over the human behavior. However, each constraint has a different effect according to the situation considered. In the particular case of virtual spaces where human interactions are mediated by computing devices such as the Internet or pervasive computing environments, the architecture is fundamental. In such environments, architectural constraints imposed by the real world no longer exist. People can see through walls using ubiquitous video cameras, they can remember not only everything they did but also everything their friends and enemies did and when, and they can follow people on the streets in real time without leaving their houses. The only architectural constraints that exist in virtual environments are those imposed by its code, or as formulated by [Mit95] "... on the electronic frontier, code is the law".

Computer scientists, who play a crucial role in the creation of those virtual environments, have the moral responsibility of ensuring that the environment's architecture will respect the fundamental principles that are the base of our society. As Lessig puts it: "We can build, or architect, or code cyberspace to protect values that we believe are fundamental. Or we can build, or architect, or code cyberspace to allow those values to disappear. There is no middle ground" [Les06]. Computing architectures must provide the tools to respect those rights and enable individuals to choose if they want to waive those rights in the virtual world, just as they do in the real world.

This issue is even more critical in future computing environments where computers permeate a greater number of human interactions. In this thesis, we study the particular case of pervasive computing systems and how the pervasive software architecture affects the way applications handle private user data. In the next sections we discuss the effects of pervasive computing on privacy, typical architectures for pervasive computing, and the privacy impact of the service oriented architecture, which is a compelling architectural style for implementing pervasive computing systems.

1.2 Pervasive Computing and Privacy

Pervasive computing, also called ubiquitous computing [Sat01], refers to a vision of future computing environments first described by Mark Weiser in his 1991 influential paper *The Computer for the 21st Century* [Wei91]. In this article, Weiser affirms that existing computers at that time monopolized user attention, while other established technologies such as writing and electricity were so well-integrated to our lives that humans could use them without paying attention to the technology itself. Weiser aimed at creating a new form

of computer, one that would require less attention from the user, that would be naturally integrated to the user's everyday life and that could be available anywhere, as ubiquitous as the air. A pervasive environment, thus, is a place that features this type of widespread computers and that supports the deployment of pervasive computing applications.

This vision created a new area in Computer Science dedicated to study the issues that arise in realizing this vision. These issues involve topics such as Networking (e.g., the development of efficient wireless protocols), Human-Computer Interaction (e.g., the design of user interfaces not based on traditional mouse, screen and keyboard), Hardware (e.g., the creation of compact devices with long battery lives), Security (e.g., how to provide authentication and confidentiality in ad hoc environments) and Distributed Systems (e.g., how to manage device mobility to facilitate application development).

One issue that attracted the attention of researchers since the first experiments with Ubiquitous Computing at the Xerox Palo Alto Research Center (PARC) is the privacy impact of the pervasive use of computers in our everyday lives [WGB99]. In what became later known as the earliest ubiquitous computing system ever designed, researchers at Xerox PARC created devices with three different sizes: a palm-sized computer (ParcTab), an electronic notebook (ParcPad) and an electronic whiteboard (Liveboard) [WSA+96]. Rooms were equipped with infrared transceivers connected to available workstations and users were encouraged to carry their ParcTabs all the time. This hardware infrastructure enabled the location of users inside the building, and supported the creation of location-based applications such as forwarding phone calls to the phone closest to a user or locating coworkers inside the building. Even though the initial design for such applications relied on a central server where all location data was gathered and which anyone could consult, it progressively changed to an architecture where clients could restrict access to their location information and eventually to a fully distributed system where each user stored his location data on his personal computer [Wei93].

In the following years, privacy was consistently mentioned as one of the main challenges in pervasive computing. Some of the aspects emphasized in the literature are how to balance seamless usage of pervasive environments but still alert users of situations when there is a potential privacy risk [Sat01]; how to enable users to control their visibility in pervasive spaces as they normally do in physical spaces (for instance by hiding from windows) [AM00]; how to manage information sharing when the boundaries between public and private spaces blur [LLR05]; how to convince the user that the pervasive environment protects his privacy [SC03] and how an individual can assure the privacy of data sensed about him [Ran04].

There are mainly three reasons why privacy is a particularly important issue in pervasive computing. First, in pervasive environments computational devices such as electronic sensors and video cameras are spread everywhere. As a result, it is easier to collect data about users of those spaces. Second, the main goal of pervasive computing is to integrate computers to user's everyday lives, making those devices disappear in the environment. As a consequence, users do not know when the system is collecting their personal information. Finally, pervasive environments must be intelligent and require little user attention. To achieve this goal, pervasive systems must extensively collect data about users to learn

their typical activities and preferences. The combination of widely available devices that are integrated into the environment and that extensively collect user-related information can potentially transform a pervasive system into a mass surveillance tool.

With the evolution of pervasive computing, its potential effects on privacy became clearer and researchers acquired a better understanding on how to build privacy-aware pervasive applications. For instance, experience showed that privacy protection is more effective when privacy is considered early on the application design phase than if it is implemented as a corrective measure after deployment of a privacy invasive application. The european "Disappearing Computer" initiative proposed high level guidelines for the design of pervasive applications [LJ04], inspired by the OECD guidelines for systems that handle personal data created in 1980 [Org80]. Langheinrich proposed a set of principles and guidelines for the design of privacy-aware pervasive applications much closer to the real world software design process and thus easier to follow [Lan01]. Hong *et al.* proposed a complementary approach based on risk analysis to identify the privacy risks of a pervasive application and how to manage those risks in the application design [HNLL04].

Based on the works above, Chung *et al.* created a set of design patterns for the development of ubiquitous computing applications, including specific patterns to deal with privacy [CHL+04]. Even though the patterns were supposed to be closer to the situations faced by application designers, hence simpler to use, in the evaluation they performed software designers did not use the particular patterns created to deal with privacy issues. This result evidences the difficulty of applying high level recommendations on real world design situations, partially due to the fact that such recommendations usually assume that pervasive application designers are in control of the whole design process and can decide how each software component will handle privacy sensitive situations.

In pervasive computing, however, this is hardly ever the case. Pervasive applications require the integration of multiple software layers, usually developed independently and uncoupled from layers above and below. Even though this is the natural approach for the rapid development of distributed applications that leverage general-purpose computing devices, it also complicates development of privacy-respectful applications since new vectors for privacy invasion appear from the interaction among those software layers. To enable effective privacy protection in pervasive computing applications, those vectors must be identified and mechanisms must be created to minimize their effect on user privacy.

In Section 1.1.4 we discussed the role of architectures for privacy protection based on the perspective of professor Lessig who, being a Law professor, uses the term architecture in a sense slightly different than traditionally used in Computer Science. The pervasive computing architecture in the sense used by professor Lessig comprises many different layers. First, the physical architecture – meaning the way computing devices are installed in a pervasive computing environment – can favor some applications of the technology and restrict others. For instance, a big screen installed in the middle of an open space have different applications than a small screen installed on the back of each chair on a theater. After that, the hardware architecture, or its design, can also have an effect over how people interact with pervasive computing. If those screens are touch screens, they allow some forms of interaction that are not possible with screens that are not sensible to

the user touch.

Finally, the software used by a pervasive environment can enable some functionalities and disable others, or simplify the development of some types of applications while complicating others. The software layer itself is composed of the software embedded on pervasive devices, the operating system that is installed on each device to manage its resources, the common middleware layer that abstracts low level details and provides higher level services to the layer above, and the applications that use middleware services to provide users with some functionality. Figure 1.4 shows how those layers relate.

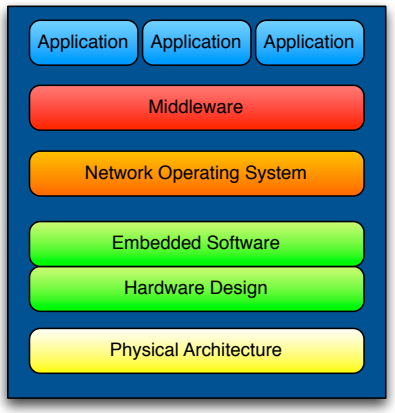


Figure 1.4: The Architectural Layers of a Pervasive Computing Environment

In this thesis, we specifically discuss the impact of the middleware on the privacy of users of pervasive computing applications for two reasons. First, since the middleware provides an homogeneous layer that is used by every pervasive application, any privacy attack targeted at the middleware directly affects users of every application. This is opposed to attacks to lower software layers, that cause an impact only to users of a particular operating system or of a particular device. In this sense, the middleware resistance to privacy attacks is more critical than resistance of the layers below. The second reason is that since the middleware mediates interactions between applications and lower level software layers, it can minimize the impact of attacks targeted at the operating system or the device's embedded software by avoiding that the results of these attacks reach the applications. The middleware has also the role of effectively creating a privacy protection layer to the heterogeneous technologies in the layers below.

In the next sections we detail the role of middleware in the development of distributed systems and its relation to privacy protection in Section 1.2.1. We motivate the use of a particular middleware for pervasive computing, namely a service-oriented middleware in Section 1.2.2, and discuss the impact that utilization of this middleware can cause on user privacy in Section 1.2.3. This issue is particularly important since the impact does not depend on the application: any application using middleware-provided functionalities is exposed to the privacy vulnerabilities of the middleware layer. Finally, in Section 1.2.4,

we define the specific pervasive environment considered in this thesis, which is used by all chapters that follow.

1.2.1 Middleware for Pervasive Computing

Distributed systems leverage the high number of computing devices and explore network connectivity to build applications that are highly scalable and extremely powerful at a comparatively small cost. The cost reduction of computers with small computing power and the evolution of network connections led to the popularization of distributed systems, and distributed applications are commonplace today, ranging from multimedia distribution systems used to manage and disseminate video and music inside a house up to systems involving millions of computers distributed worldwide such as the Internet.

Throughout the evolution of distributed systems, middleware played an essential role. Middleware is a software layer located in between network operating systems and application components [Emm00a]. Middleware allows the rapid prototype and deployment of distributed applications by leveraging existing software and hardware components, which is a much more cost-effective and flexible solution to the alternative of designing and creating a new software and hardware infrastructure to support deployment of a distributed application.

The main goal of a middleware, thus, is to create a homogenous layer that handles the heterogeneity of existing components in the lower layers, while providing abstractions that facilitate the task of developing distributed applications to the upper layers. The focus given to these abstractions distinguish existing middleware solutions. Transactional middleware, for instance, concentrates on how to provide applications with a simple abstraction to perform distributed transactions, while the main goal of a message-oriented middleware is to provide an abstraction that facilitates message exchange between components [Emm00b].

Simultaneously to the middleware evolution, miniaturization of computing components and the development of wireless network technologies led to the creation of a new type of distributed computing paradigm called mobile computing. This new paradigm is much more chaotic and unstable than traditional distributed systems [RPM00]. While computing nodes were traditionally assumed to be connected through fixed, homogeneous, and relatively stable networks, in mobile computing a network infrastructure does not exist and nodes opportunistically create connections to each other based on their physical proximity. Disconnections are the norm instead of the exception and local contextual information must be determined at run time by mobile nodes, while fixed nodes could usually be configured in advance [FZ94].

Pervasive computing environments result from the combination of highly distributed systems with mobile computing environments. In a sense, pervasive computing subsumes mobile computing and goes further, and thus pervasive systems must face not only the typical issues of mobile computing but additional difficulties created by some features of pervasive computing such as the invisibility of computing devices and how to support scalability of local interactions [Sat01]. Pervasive computing can thus be considered as the

next step on the evolution of distributed systems.

Not surprisingly, middleware designed for traditional distributed systems fails to address pervasive and mobile computing requirements. Classic middleware solutions are heavy since they assume resource-rich nodes, they suppose nodes directly connected to each other and thus mainly rely on synchronous connections, and are designed for static nodes where contextual information is mostly constant and requires less adaptation from the middleware.

A compelling model for middleware design in pervasive computing is reflective middleware [RKC01]. Reflective middleware maintains an explicit representation of its internal structure that allows systems and applications to inspect it and reconfigure it if needed [KCBC02]. The middleware itself can use reflection to self-adapt to changes in the environment [CBM+02]. Through reflection, middleware can be lighter, use available resources more efficiently and better adapt to context.

Novel middleware solutions were proposed based on the requirements of pervasive and mobile computing offering mechanisms to facilitate adaptation to the environment dynamics. Gaia [RHC+02], for instance, uses reflection to better adapt to resource heterogeneity and dynamics while one.world [Gri04] report all changes from the environment to applications providing hooks to application-initiated adaptation. However, application of the above pervasive middleware solutions remains restricted.

1.2.2 Service-oriented Pervasive Computing

Recently, the need to connect applications across enterprise boundaries led to the introduction of Service-oriented Computing (SOC), a computing paradigm based on the Service-oriented Architecture (SOA) that uses services as basic elements for application development. Services are self-describing and open components that support rapid composition of distributed applications [PG03]. A service-oriented architecture enables the interaction between clients and service providers through functionalities that facilitate service discovery, composition and access. Figure 1.5 shows the layers of an SOA.

The basic features that allow clients to consume services are part of the lower Basic layer. They include mechanisms for service publication, discovery, filter, and binding. The layer above, Service Composition, provides the features necessary to create composite services from basic services. Additional concerns that arise when services are combined in composite services such as the composition quality of service (QoS) and the coordination among composed services are addressed at this layer. Finally, the Service Management layer provides functionalities to help service operators to manage the service infrastructure.

The SOA is supported by a middleware that implements the features described above. A service-oriented middleware connects heterogeneous components and systems, enables access to services, and offers abstractions that facilitate publication, description, discovery and binding of services [PTDL07]. Other more advanced service-related functionalities such as service composition can also be implemented at the middleware layer to simplify development of applications that require the combination of services to perform a certain task [Ben07, ZBN+04]. Such middleware must handle multiple layers of heterogeneity

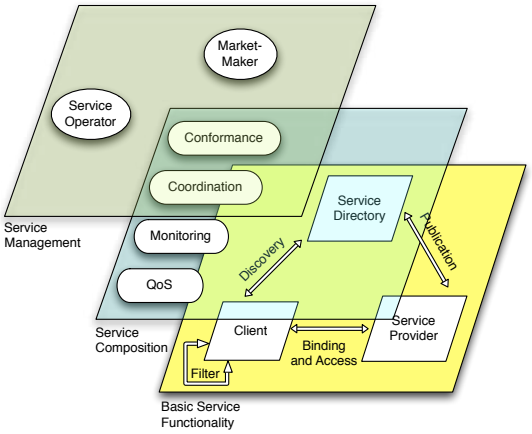


Figure 1.5: The Fundamental Layers of an SOA

including hardware, operating system, communication protocols and even other middleware solutions, working actually as a middleware for middleware [Vin02]. In summary, a service-oriented middleware overcomes heterogeneity to provide protocols and abstractions that facilitate service discovery, composition and access.

Towards the goal of creating a service-oriented middleware, an approach widely adopted is to use Web technologies. These technologies are present on a number of environments and platforms, and provide the benefits of creating a common layer independent from operating system or network technologies, and constitute a compelling platform for creation of a service-oriented middleware. The instantiation of a service-oriented architecture based on Web technologies is called a Web Service Architecture [CDK+02]. Web Services leverage the Internet infrastructure and the ubiquity of Web-related protocols and technologies to provide a concrete architecture for the large-scale deployment of services.

At the core of the Web Services technologies are three specifications. The Simple Object Access Protocol (SOAP) [W3C07] is an XML-based protocol for communication that defines a standard message format through which services can exchange data and call each other methods. The Universal Description, Discovery and Integration (UDDI) specification [OAS05d] specifies the implementation of a service repository, defining the data it contains and the protocol to access it. Finally, the Web Services Description Language (WSDL) [W3C01] defines a standard language to describe Web Services, including their interface and access messages. Based on those standards, other standards were created to define different aspects of services such as security[9] and reliability[10]. Figure 1.6 shows the typical Web Services architecture.

Given the benefits of the Service-Oriented Architecture and the wide availability of Web technologies, even in resource-constrained devices, Web Services are a powerful approach for the realization of pervasive computing [IST+05]. Indeed, most current mobile devices

[9] www.oasis-open.org/committees/wss

[10] www.oasis-open.org/committees/wsrm

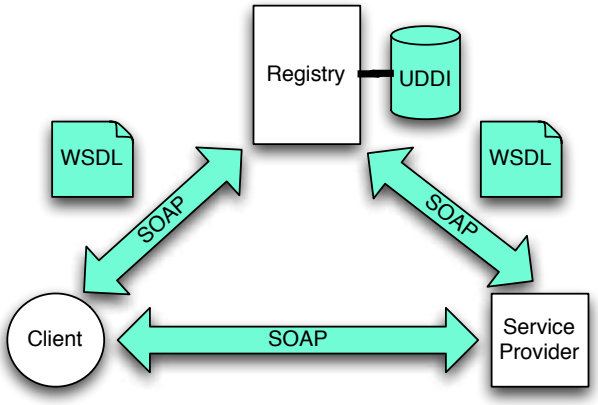


Figure 1.6: The Web Services Architecture

feature Web browsers and today there are lightweight Web servers specifically designed to run on devices with constrained hardware resources, such as Nokia's Mobile Web Server[11] as well as SOAP engines designed for enabling SOAP message processing in devices with limited processing power and memory[12].

Traditional service-oriented middleware, however, is not suitable to pervasive computing requirements. Pervasive-specific challenges such as user mobility, context-awareness, energy management, privacy and trust [Sat01] are much more important in pervasive computing than in Web environments and thus are not tackled by common Web service middleware. Fortunately, initiatives such as the PLASTIC middleware[13] aim at providing additional abstractions to a service-oriented middleware for pervasive computing so as to enable the successful realization of service-oriented pervasive computing.

1.2.3 Privacy Issues in Service-oriented Pervasive Computing

The evolution of middleware to support the flexibility and mobility requirements of pervasive computing has neglected some important characteristics of those new distributed computing paradigms that differentiate them from more static environments. In traditional distributed computing, the infrastructure connecting the nodes is purpose-specific, fixed and can be trusted: networking devices can be installed in secure rooms to protect them from physical unauthorized access and only the system's administrator is able to access them. Network connectivity is stable, and sensitive interactions between components can rely on an always available trusted third-party. Furthermore nodes are fixed, their context hardly ever changes, and thus it is easier to protect them from attacks and to

[11] research.nokia.com/research/projects/mobile-web-server

[12] csoap.sourceforge.net

[13] www.ist-plastic.org

identify who is accountable in cases where information is misused. Finally, in traditional distributed systems the interaction between users and computers is clear and most of the time users can identify when their personal data is going to be used by the system.

In pervasive computing, however, these assumptions no longer hold. First, ad hoc networks are an integral part of pervasive computing environments, and since they opportunistically create a temporary structure that utilize user devices to communicate, the network infrastructure can no longer be trusted. Network connectivity in pervasive computing is intermittent and it is difficult to ensure that a trusted third-party is available to mediate sensitive interactions. Users in pervasive computing carry portable devices that are harder to protect, and interact with other untrusted portable devices that may be only temporarily available. In such conditions, enforcing data usage rules is extremely difficult as well as ensuring accountability when usage rules are not respected. Lastly, as computers in pervasive environments "disappear" [Wei91], users are less aware of computer interactions and the natural instinct of protecting information disclosure fails to work in such cases. As an example, individuals adapt their behavior when they are close to windows but the human brain fails to do the same in front of a surveillance camera.

These properties of pervasive computing have a particularly destructive effect over the user privacy. Since the pervasive network infrastructure can no longer be trusted, data transferred from clients to service providers and vice-versa must be protected during transmission to avoid undue access to data. Protection mechanisms that require a trusted third-party are only of limited utility in pervasive computing since they may not be reachable and thus must be adapted to the pervasive setting. Since accountability is harder to obtain in mobile settings, users must be able to minimize information disclosure to reduce the effects of malicious usage of personal information. These issues are aggravated by the fact that computer interactions in pervasive are unobtrusive, and a higher quantity of personal information is disseminated in pervasive systems than in traditional distributed computing.

In the specific case of service-oriented pervasive computing, the abstractions that enable interaction among the three components depicted in Figure 1.5 must be designed considering the particular requirements of pervasive computing. Our analysis of the privacy impact caused by service oriented features for enabling service provision and consumption in pervasive computing suggests that existing mechanisms for service access, discovery and composition found in current service oriented middleware introduce new forms of privacy invasion when applied to pervasive computing.

Service access in pervasive computing has particular characteristics. Pervasive computing environments are cooperative and ad hoc, and users provide services to each other without necessarily accessing a fixed infrastructure. In such scenarios, public key encryption is undesirable because it requires access to to a fixed infrastructure for certificate validation and also because it has the side-effect of associating a real world identity to every transaction, even when authentication is not required.

Regarding service discovery, there are two main issues: first, service discovery data is sensitive and service discovery protocols traditionally assume a trusted service directory. The directory handles service advertisements and service discovery requests, so clients and

service providers must trust it to not misuse that information. However, in pervasive computing it is not feasible to have trusted service directories because the administrative requirements are incompatible with open systems and ad hoc networks. The second issue concerns service directory federation. Common protocols for service discovery through different administrative domains normally create a federation of service directories that are mutually trusted, and forward service requests through the directory hierarchy. This approach is problematic in pervasive computing because the connections among administrative domains are usually temporary and opportunistically created, and mutual trust cannot be assumed. As a result, the contents of user-defined service discovery requests may be unnecessarily exposed to malicious nodes on the environment.

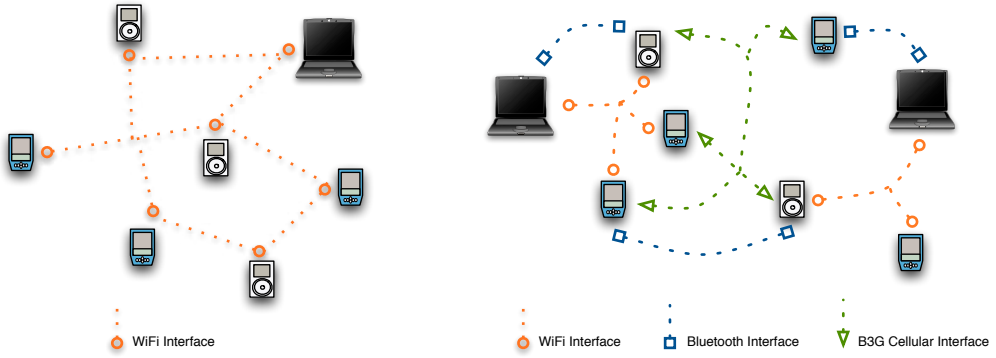
Service composition can also make users loose control of their personal data and become victims of privacy invasions. Service composition enables users to benefit from the full potential of pervasive computing systems by combining existing services into new unpredicted applications. However, in such settings hardly ever the exact user-defined composition can be executed using available services. Usually the middleware recomposes available services to obtain the same functionality as defined by the user composition. This approach increases service composition flexibility, but also modifies the user-defined task distribution into service providers, which may allow providers to correlate data in ways the user does not expect.

1.2.4 Pervasive Network Specification

In the context of this research, we consider a certain type of pervasive computing environment that we believe will become popular in the near future due to the popularization of next generation mobile computing devices. Today, individuals carry various wireless-enabled multi-purpose digital gadgets, and pervasive computing environments will explore not only their cooperative possibilities but also their integration to the fixed infrastructure.

As a result, we view pervasive computing as the opportunistic integration of fixed, infra-structured networks with mobile ad-hoc networks (MANETs). For instance, a group of collocated resources (e.g., a printer and a projector) may be connected through a Bluetooth ad hoc network while another remote resource is only accessible through a WiFi infrastructure (e.g., a file server). The convergence of multiple network interfaces into a single mobile user device enables the connection of those two isolated systems, overcoming network heterogeneity and enabling users to access a greater number of resources to perform their tasks. Current cellphones featuring heterogeneous network interfaces (e.g., Wi-Fi, Bluetooth and cellular 3G) can provide users with access not only to resources on networks directly connected to the device but also by forming ad hoc networks and accessing resources on remote networks through other mobile devices.

We call these mobile networks that use heterogeneous wireless technologies *hybrid mobile ad hoc networks* (HMANETs). They are typically formed by independently managed networks connected to each other by multi-homed devices (also referred to as *bridges*). All nodes in such networks are potentially mobile, as opposed to wireless mesh networks [AWW05] where some nodes are assumed to be fixed (mesh routers) while others



(a) MANET (b) HMANET

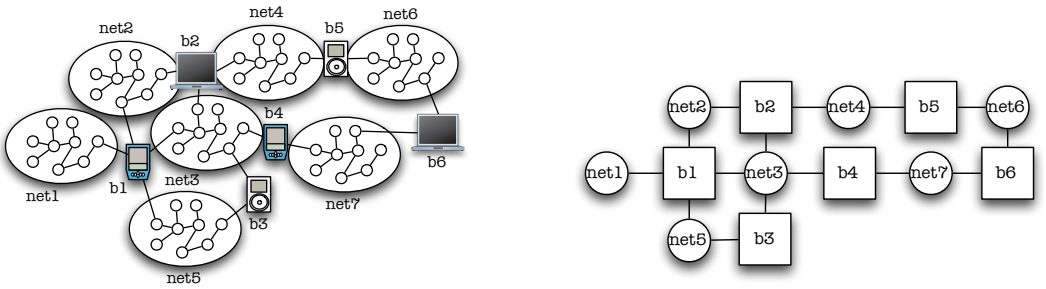
Figure 1.7: Differences between a MANET and an HMANET

may move (mesh clients). In HMANETs, also, any two nodes may be directly connected by multiple different links in opposition to traditional mobile ad hoc networks (MANETs) where any two nodes share at most one direct connection. More importantly, those connections use different technologies and present heterogeneous properties such as delay, throughput, security, energy consumption and cost. Figure 1.7 shows the differences between MANETs and HMANETs.

We assume that all networks run the IP protocol, but global IP routing is not enabled. In other words, each network is independently and autonomously organized: the same addresses may be used by different networks and networks do not know each other *a priori*. Devices featuring multiple network interfaces can volunteer to act as bridges and transfer packets among networks, creating an overlay network formed by bridges and heterogeneous networks. The process through which a regular node is elected as a bridge is out of the scope of this thesis. Here, we assume that such an election algorithm exists. A route in an HMANET, then, alternates between bridges running different software platforms and networks using different technologies.

This scenario, where network characteristics must be taken into account for routing, is better represented by a bipartite graph $G = (U \cup V, E)$, where $\forall e = (i, j) \in E, i \in U$ and $j \in V$. In other words, U is a set of nodes representing networks, V represents mobile devices and every edge in E connects a network node to a device node. A route, in this graph, is an ordered set that alternates elements in U and elements in V. Figure 1.8 shows an example of this representation.

With regards to the software architecture, we adopt a service-oriented architectural style where resources and applications are modeled as services and communicate their interfaces to potential clients through service directories. A mobile service-oriented middleware offers the required mechanisms for service access, discovery and composition. The roles of client and service provider are not fixed: a user can be a client in one interaction and a service provider in the next interaction. Users can even play both roles at the same



(a) An HMANET (b) Corresponding Bipartite Graph

Figure 1.8: A Bipartite Graph Representation of an HMANET

time, consuming and providing services simultaneously.

We adopt a service model where a service is characterized by its inputs, its outputs and its properties. A property is anything that helps to describe a service besides its inputs and outputs, such as a service category, a price or other quality parameter such as latency. To enable users to effectively leverage the potentially large quantity of services available, the middleware also provides a flexible service composition mechanism that automatically re-composes accessible services to realize a user-defined composite service.

Users in a given network are able to discover and access not only services in the local network but also in remote networks that are dynamically composed. Service directories forward service discovery requests to distant directories, augmenting the set of services available to clients. Remote networks dynamically join the system and nodes in different networks do not know each other in advance. For this reason, it is not likely that clients and service providers agree on a service description, and to increase the probability of finding a specific service among the possibly numerous available services, service discovery supports both syntactic and semantic-annotated service descriptions. Service directories must also be always accessible and thus we assume that service directories can be dynamically deployed on user devices to enhance service discovery availability.

We also consider that the middleware provides a flexible mechanism for service composition that, based on the user-provided composition workflow, suggests possible executable workflows that realize the user task using existing services of the environment. This mechanism is also necessary because of the environment openness. As users do not know beforehand which services will be available in an environment, the user-defined composition probably contains services that do not exist in the user current location. This composition mechanism enables the user to realize his task even if the exact services that the user expect are not available.

A Trust Model for HMANETS The environments described above augments the set of accessible services for a client by enabling the opportunistic consumption of services deployed in remote networks. Consumption of remote services, thus, requires that a user discloses service discovery and access data to nodes in networks he does not know and

that he does not trust. Devices in remote networks may not have any reason to fairly use clients personal information or may even join the environment only to abuse personal data available through the platform.

Disclosure of service discovery and service access data to untrusted entities is critical for privacy because this data is sensitive and malicious nodes can use it to attack the individual's privacy. Service discovery data may reveal the user's desires and intentions, or can be used to link consecutive requests. Service access data can include identifiable information (e.g., name or telephone) or sensitive information (e.g., list of diseases), and can also reveal information that the user does not intend to disclose, specially when stored and analyzed over a period of time. To better evaluate the privacy risks involved when a service-oriented architecture is used to support application development and deployment in an HMANET, it is necessary to define a trust model consistent with multi-network pervasive computing.

Consider a pervasive environment comprising n networks $N_1, ..., N_n$. Each network has a single service directory D_i responsible for handling service discovery requests from all local clients belonging to network N_i. Two distinct networks N_i and N_j can be connected by a bridge B_{ij}, in which case requests handled by D_i can be forwarded to D_j through B_{ij}. If a bridge connects more than two networks, for instance networks i, j and k, we consider the bridge B_{ijk} as multiple bridges connecting two networks (e.g., B_{ij}, B_{jk} and B_{ik}). Figure 1.9 shows this configuration. Our trust model is based on the following assumptions:

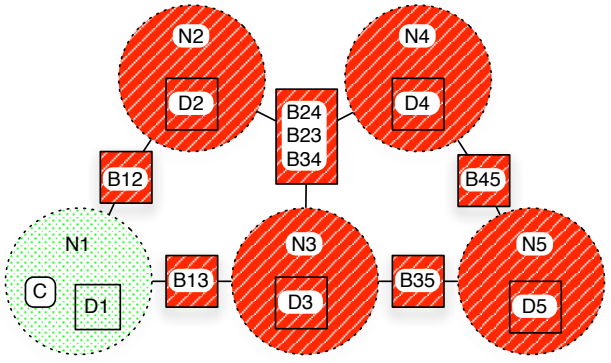


Figure 1.9: A Trust Model for HMANETs

1. Clients in network N_i trust the local directory D_i to fairly use their personal information. As clients and the directory are located in the same network, their trust relationship can be created based on a contract that defines how the directory can use client information (and possible countermeasures in case of abuse), on the client's personal experience when using directory D_i, or on any other trust establishment protocol. In Fig. 1.9, client C trusts directory D_1 to fairly use its personal information.

2. Clients in network N_i do not trust any bridge $B_{kl}, \forall k, l \in \{1, ..., n\}, k \neq l$ to fairly use their personal information because they are not directly responsible for bridge election and their requests are forwarded to bridges regardless of their trust judgment. Besides that, clients do not know critical information about the networks connected by the bridges such as topology or network technology in place, which are important to support the clients' trust decisions. For instance, a client may not trust WiFi networks using weak encryption algorithms. In Fig. 1.9, client C does not trust bridge B_{12} to fairly use its personal information.

3. Clients in network N_i do not trust any other directory D_j, $j \neq i$ to fairly use their personal information. As clients do not interact directly with remote directories and their association is transient, it creates additional obstacles for trust establishment protocols (for instance protocols based on past experiences). In Fig. 1.9, client C does not trust directory D_2 to fairly use its personal information.

4. All the entities in the environment trust that bridges and directories will correctly execute their protocols. We assume that every directory can determine if bridges are running the right software version and vice-versa. We also suppose that one component can detect another component misbehavior, and exclude ill-behaved components from the platform in later interactions. We are not investigating issues that arise when components maliciously run their protocols.

1.3 Document Organization

This work focuses on privacy protection in pervasive computing. Our thesis is that the software architecture of a pervasive environment has a fundamental impact on the user behavior in what concerns privacy. Clearly, effective privacy protection in pervasive computing requires measures that influence the four constraints that regulate human behavior, described in Section 1.1.4. It requires laws to punish companies or governments that surreptitiously use pervasive computing to invade the privacy of consumers and citizens. It requires market incentives to systems that protect user privacy higher than the incentives to invade it, for instance users boycotting privacy invasive services. It requires individuals to develop a privacy mindset which allows them to perform informed decisions about privacy.

The software architecture, however, plays a more important role in enabling and stimulating the creation of privacy-aware pervasive applications or preventing and inhibiting development of privacy invasive software for pervasive computing. In such environments, where computers mediate the interactions between individuals and augment human abilities, the software is responsible for imposing a number of restrictions that exist in the real world but that disappear in the virtual world.

Towards this objective, we study the particular case of using service-oriented architectures in pervasive computing, and the resulting architectural risks to privacy that appear. To mitigate those risks, we propose a privacy-aware service-oriented middleware for per-

vasive computing that protects users from privacy attacks in service-oriented pervasive computing. Applications deployed on top of this middleware can provide users with more mechanisms to control private information disclosure and thus expose them to less possibilities for privacy invasion.

This document is organized as follows: in Chapter 2 we discuss relevant work on the subject of privacy protection in software systems and applications. Surprisingly enough, the study of privacy-aware computing architectures is not the primary focus of the research community. In Chapter 2 we review relevant work in privacy research and categorize them according to the protection they provide against privacy attacks in service-oriented pervasive computing. In Chapter 2 we specifically review research related with privacy, but during this thesis we were inspired and used results from works in other areas. As those works form a very heterogeneous group, for the sake of clarity they are presented in the chapters where they are most relevant.

After motivating and positioning our work according to existing research, we introduce our main contribution: a service-oriented middleware architecture for privacy respectful pervasive computing that features mechanisms to avoid privacy attacks directed to the service-oriented architecture. We first present components for protecting privacy during service access in Chapter 3, during service discovery in Chapter 4 and during service composition in Chapter 5. In Chapter 6 we use an application scenario to describe how these components cooperate in a middleware architecture to increase privacy during service provision and consumption.

Chapter 3 discusses how the multiplicity of paths between two nodes in an HMANET can be used to improve privacy in service access. We specify an hybrid ad hoc routing protocol that provides clients with multiple paths to a destination on-demand, while keeping shortest routes to all destinations pro-actively, reducing the overhead of multipath route discovery only to situations when it is required. Based on those paths, we describe a protocol for route selection that attempts to discover routes resistant to compromise of nodes running the same software platform or compromise of network connections using a given technology. Finally, a message sharing protocol uses this set of routes to distribute message shares that have minimal size overhead given the user-required reliability.

Chapter 4 details our privacy-enhanced protocol for service discovery in pervasive computing. It supports semantic and syntactic service discovery, reduces the sensitivity of service descriptions and enables deployment of service directories in untrusted devices, an important requirement in pervasive computing. Our protocol creates generalized versions of service descriptions that can potentially represent a set of services instead of a single service. Directories use privacy-enhanced service descriptions instead of regular descriptions for service matching, and are unable to identify which exact service a client is searching for or a provider is announcing. The protocol also provides mechanisms to increase privacy protection in distributed service discovery, when directories forward service requests to remote directories in untrusted networks.

Chapter 5 presents our model for analyzing the privacy impact of service compositions that are structurally different but functionally equivalent to user-defined compositions. When users create composite services, the task partition among service providers in the

abstract workflow defines an implicit data flow that, if not respected, can lead to privacy invasions. Instead of restricting composition only to executable workflows that have the same task partition as the abstract workflow, which would be too constraining in open environments like pervasive computing, we introduce a model that enables users to reason about the privacy impact of executing workflows with different task partitions but that are functionally equivalent. With this model, users can select compositions that realize the required task and minimize the effects on their privacy.

After detailing the middleware components in the chapters presented above, we introduce a scenario in Chapter 6 that allows us to introduce the middleware architecture and to describe how the middleware components cooperate to improve privacy in service-oriented pervasive computing. The scenario describes a fictional application called "Cupd", a mobile dating service that motivates privacy issues in service access, discovery, composition. We show how the middleware components in the user's device cooperate with the middleware on bridges to minimize the impact of each of those issues.

We expect that after reading the chapters described above it will become clear that privacy is an important issue in present societies but also that privacy protection in information systems is a complex problem. In this thesis we contribute to the effort of creating computer systems that respect the privacy of their users, but much remains to be done for this vision to become a reality. In Chapter 7 we propose some perspectives for the future exploitation of the work presented in this thesis and also other ideas to improve privacy protection in information systems, as well as our final conclusions and remarks.

P-r-i-v-a-c-y
Is priceless to me
P-r-i-v-a-c-y

Pearl Jam, "Pry, To"

2

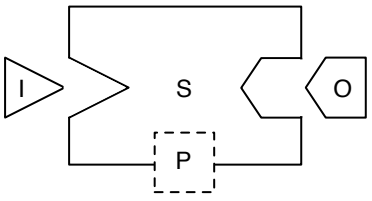
Privacy in Service-Oriented Pervasive Computing: State of the Art

Since the first experiments, privacy is considered an important issue in pervasive computing. But privacy is not a concern specific of pervasive computing research: given the particular impact of computers on privacy described in Chapter 1, other areas of computer science research such as databases and networks also have been developing mechanisms for privacy protection in specific scenarios and applications. The Internet popularization increased awareness on the risks that computers pose to privacy, and users started to develop techniques to improve anonymity on the Web. That is how the first privacy-enhancing technologies (PETs) were born [GWB97].

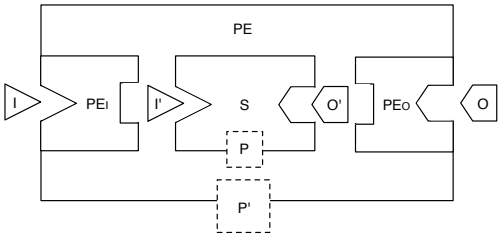
The evolution of privacy research in parallel among different areas produced a number of interesting results but also created some problems: it is not clear how these works relate to each other and researchers disagree on the meaning of some terms. Notably, many works interchangeably use the expressions *privacy-enhanced* and *privacy-aware* when referring to mechanisms whose purpose is to improve the privacy level of users of a system or of a technology. In our view both terms convey different meanings and ideas, and their indiscriminate use can be confusing.

In the context of this thesis, a **privacy-enhanced** system (protocol, or technology) is one where privacy concerns appeared as an afterthought, and privacy protection is included as an additional layer. This layer does not fundamentally change the original system and can even be optional; for instance implemented in the form of a plug-in. Consider a system S that generates outputs O when supplied with inputs I providing privacy P. A privacy-enhanced version of S, that we call PE, introduces two optional systems PE_I and PE_O that respectively transform the inputs supplied to S from I to I' and the outputs generated by S from O' to O such that the system PE provides privacy P', with $P' > P$.

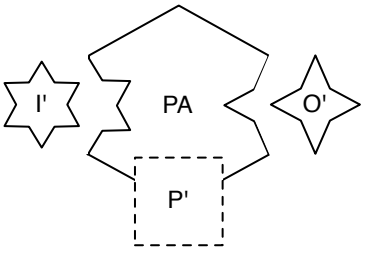
On the other hand, a **privacy-aware** system (protocol, or technology) is designed with privacy in mind. Its design can be based on an existing system, but adding privacy-awareness causes such structural changes to the original system that privacy protection becomes a system property instead of an optional feature, and the original system may have to be replaced by its privacy-aware version. If we consider again a system S that generates outputs O when supplied with inputs I providing privacy P, its privacy-aware version PA uses inputs I', with $I' \neq I$, to generate output O', $O' \neq O$, with privacy P', $P' > P$. Systems S and PA, however, are incompatible: their inputs and outputs are not interchangeable and they have unrelated requirements.



(a) Original System



(b) Privacy-Enhanced Version



(c) Privacy-Aware Version

Figure 2.1: Representation of a System, its Privacy-Enhanced and Privacy-Aware Versions

These definitions do not imply a quality judgement. Neither privacy-aware systems are better than privacy-enhanced systems nor the opposite. In some situations a privacy-aware system may provide a stronger protection than a privacy-enhanced equivalent since it was designed having privacy as a requisite, but it may also incur a bigger overhead since the mechanisms for privacy protection affect all users and cannot be easily disabled.

In some cases privacy protection will be more appropriate if implemented as a privacy-enhancement, while in other cases effective privacy protection will require the system to be more radically modified to become privacy-aware.

In this chapter we clarify the relationship among existing privacy technologies when applied to service-oriented pervasive computing. We present a categorization of possible strategies for privacy invasions in service-oriented systems and propose to classify privacy technologies according to their protection against these strategies. We consider the basic features provided by a service-oriented middleware, namely service access, discovery and composition, analyze how existing results can be used to increase privacy protection of each of those features and identify their shortcomings when applied to pervasive computing. This categorization was initially proposed in [CI07] and it is extended here.

2.1 A Categorization of Privacy Invasion Strategies

Results in privacy research form a very heterogeneous set. Some techniques are designed to solve a very specific privacy issue of a recently introduced technology while others are more generic and can solve a set of issues in different situations. Some techniques can be deployed on top of invasive technologies while others require modifications to the original system. Some techniques can be combined to improve privacy protection, but most of the time they are incompatible with each other. This difficulty to relate existing results is an obstacle for a systems approach to privacy protection.

Pervasive computing, in particular, involves technologies from heterogeneous and unrelated areas, thus it is hard to determine how privacy protection techniques from each of those areas can cooperate to enhance user privacy. To allow for a better understanding of the goals of existing privacy-aware and privacy-enhanced systems we propose to classify them according to the protection they provide against privacy invasion strategies in service-oriented pervasive computing. In the remaining of this section we identify the entities involved in a service access and their roles with regards to privacy invasions.

A service-oriented application comprises at least two entities: a **client** and a **service provider**. The client accesses a service offered by the provider and during this access both parties may disclose personal data to each other. Many applications require some type of mediation, e.g., for security purposes, and then a **third-party** must receive (eventually private) service access data to perform the mediation. Finally, an **eavesdropper** unrelated to the access can spy on the interaction, e.g., by monitoring the communication channel. Figure 2.2 shows how clients, service providers, eavesdroppers and third parties relate. Thick lines represent required entities and interactions while dotted lines represent optional entities and interactions.

One can consider third parties simply as additional service providers and disagree with the above description representing third parties as distinct entities. When mediating an access, however, third parties obtain information from both clients and service providers. While a regular service provider have access only to information that clients disclose, third parties mediating an access obtain information from both parties, and this data

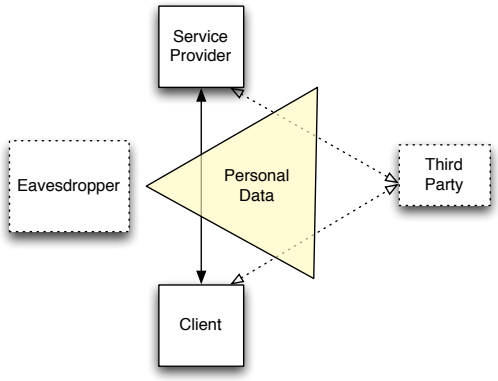


Figure 2.2: Entities and Interactions in a Service-oriented Architecture

can be correlated to infer private information about them. Treating the third party as a dissociated service provider would fail to capture this aspect.

Our definition of privacy from Section 1.1.1 includes two main aspects of privacy: **control** and **awareness**. As a result of this definition we conclude that a privacy invasion occurs when one entity loses control of personal information disclosure or when it is not aware of the effects on privacy of disclosing some data. Those situations may happen when an entity obtains improper access to user private information or when a private expectation from the user is not respected. The entity that suffers a privacy invasion is the **victim**, while those that benefit from the information disclosed in that manner are the **intruders**.

Based on the scheme of Figure 2.2 we can identify the privacy invasion victims and intruders of a service-oriented architecture. The client is the first potential victim since it is likely to disclose some private data during service access. In this case, the other three entities can be intruders: the service provider, the third party and the eavesdropper. The service provider can also be a victim, for instance, if it is providing a service that reveals personal data (e.g., a radio service that broadcasts its musics and exposes its musical preferences). In this case, the service provider can be a victim of privacy invasions from the client, from the third party and from the eavesdropper.

Third parties can also be victims of privacy invasions. Consider a reputation system where a client can ask for the opinion of another user (the third party) before consuming a service. A client using this service may enumerate all services accessed by the third party through consecutive checks for service reputation. In this case, clients, service providers and eavesdroppers can be intruders. Finally, eavesdroppers can also be victims of privacy invasions, for instance disclosing their location while monitoring a communication channel. However, eavesdroppers are undesirable entities and their privacy is not a concern of an architecture for privacy protection. Table 2.1 summarizes the potential intruders and victims of privacy invasions in service-oriented architectures.

Intruder	Victim
Client	Service Provider
	Third Party
Service Provider	Client
	Third Party
Observer	Client
	Service Provider
	Third Party
Third Party	Client
	Service Provider

Table 2.1: Intruders and Victims of Privacy Invasions

2.1.1 Strategies to Invade Privacy in Service-oriented Architectures

Having defined the potential intruders and victims of privacy invasions in service-oriented architectures, we can now analyze the strategies they may use to invade privacy. Such strategies have the common goal of either forcing the victim to lose control of private information disclosure or to break a victim's privacy expectation. We must initially identify how entities control private information disclosure and what are their privacy expectations to determine the invasion strategies that may follow.

A **privacy invasion strategy** is a technique for improperly obtaining access to private data. A **privacy attack**, on the other hand, is a method for investigating a particular aspect of one's private life. A privacy attack has a clear intent, and to perform it an attacker selects the most suitable privacy invasion strategies to achieve this goal. Here we are interested in characterizing strategies for privacy invasion in service-oriented pervasive computing.

In any interaction, unless clearly stated otherwise, clients and service providers have a number of privacy expectations and must be able to control how private data is disclosed. First, both parties expect that data disclosed in their interaction is not accessed by any unrelated entity. Second, they expect to be the only entities aware of their interaction. Third, they expect that data disclosed during the interaction is used for the interaction purpose only and fourth, they must be able to disclose the minimum amount of personal information to perform the task. This is valid for clients consuming services, but also for clients and service providers interacting with a third party.

Privacy invasion strategies aim to subvert the controls or break the expectations described above. The first invasion strategy happens when the content of a service access is disclosed to entities other than the client and the service provider. If this entity is an eavesdropper, there is clearly a privacy invasion since eavesdroppers are not related to the

service access and must not obtain data revealed during the interaction. Third parties also should not access this data, but since they mediate the interaction, they may have to obtain some of the service access data to perform their role. In this case, however, this requirement must be clearly stated to both clients and service providers.

The second strategy occurs when unrelated entities learn about the existence of a service access. This may happen, for instance, when an eavesdropper identifies the source and destination of a message, or when a third party discovers that a client wants to access a service. As with the first strategy, eavesdroppers must not have access to that information, while third parties can only have access to the data if needed to perform their roles and clearly stated to clients and service providers.

The third strategy uses data disclosed for a given purpose to other objectives, breaking the user expectation regarding data usage. This strategy concerns only data legitimately disclosed to entities involved with the interaction and that are expected to access that data. It can happen, for instance, when a service provider uses data disclosed in the context of a service access to offer another service to the client without his consent. Whenever an entity wants to use data disclosed by another entity for additional purposes it must make it explicit, and if an entity uses data for a purpose that was not allowed the data owner must be informed.

Finally, the fourth strategy aims at reducing the control that an entity has over personal information disclosure by forcing it to reveal more data than necessary for an interaction. This can happen, for example, when clients are required to identify themselves to use services that are not based on their identity. This strategy also concerns only parties directly related to an interaction and that can legitimately request data. Table 2.2 summarizes the possible privacy invasion strategies that can be performed by each of the elements involved in a service access.

Intruder	Privacy Invasion Strategy
Eavesdropper	Learn about the occurrence of a service access
	Learn about the content of a service access
Third Party	Learn about the occurrence of a service access
	Learn about the content of a service access
	Obtain more information than required to mediate a service access
	Use obtained information for a purpose different than mediation
Client	Obtain more information than required for the service access
	Use obtained information for a different purpose
Service Provider	Obtain more information than required for the service access
	Use obtained information for a different purpose

Table 2.2: Privacy Invasion Strategies

2.1.2 A Framework for the Classification of Privacy Research

We propose to classify existing results in privacy research according to the protection they provide to the privacy invasion strategies described in Section 2.1.1. This classification enables us to determine among the existing solutions for privacy protection those that are relevant in the context of SOAs and also to identify areas where more research is needed to effectively protect the privacy of entities participating on the provision and consumption of services.

To avoid the invasion strategies described in the last section, a service-oriented architecture must provide mechanisms to: (i) protect the existence of a service access; (ii) protect the content of a service access; (iii) limit personal data disclosed during a service access and (iv) control usage of personal data disclosed during a service access.

When we consider the service discovery and the service composition features, two aspects concerning privacy must be analyzed. First, the middleware can provide those features as services themselves, in which case access to these features must be protected using the mechanisms described above. Second, information disclosed during a service discovery or during a service composition concerns a potential service access. The mechanisms described above must also protect the privacy of service access from attacks targeted at service discovery and composition data.

Figure 2.3 presents a view of our proposed categories and their relation to the invasion strategies and potential intruders in SOAs. If we consider the circles from the center to the edges, in the center we see the privacy invasion strategies identified in Section 2.1.1. The next two segments relate these strategies to the potential privacy intruders in service-oriented architectures as specified by Table 2.2. The following segment shows how the invasion strategies in the center can be prevented as described in this section. These are the categories that we use to classify existing privacy research in the context of SOAs. Finally, the outermost segments display the section where each category is discussed in this chapter for the service access, service discovery and service composition features of a service-oriented middleware.

In the following sections, we classify relevant work in privacy according to this framework. The shortcomings from existing privacy protection technologies when applied to service-oriented pervasive computing motivate the research performed in this thesis, which is presented in the chapters that follow.

2.2 Privacy Research in Service Access

We start our classification focusing on the service access feature of a service-oriented middleware. This feature enables clients that already possess a reference to a service provider to interact with the service by sending and receiving messages. Some of the techniques presented here can also be used to increase privacy in service discovery and service composition: if we consider these features as services, entities using these features are simply performing a service access. However, as we will see later, service composition

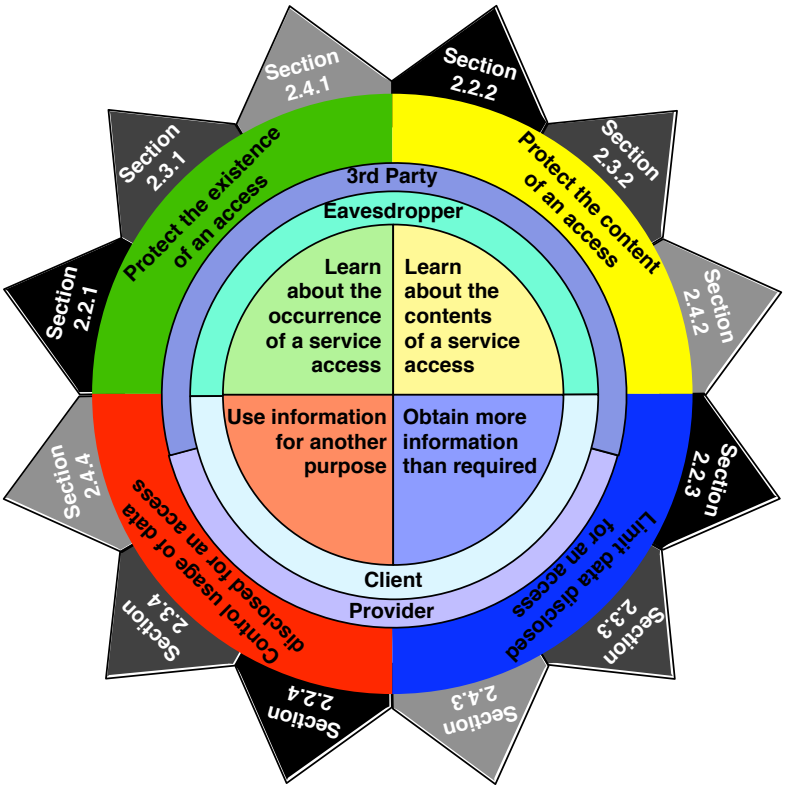


Figure 2.3: A Categorization of Privacy Research Results Applied to SOAs

and service discovery present additional requirements that call for more specific solutions.

2.2.1 Protect the Existence of an Access

Any system reveals some internal information as a normal result of its operation. For instance, the CPU usage of a server can determine if its services are being used. These data channels that were not originally designed for information transfer are called *covert channels* [Lam73]. Since these channels are not part of the original system design, data they transfer is not affected by existing access control measures. Intruders, thus, can use these channels to subvert access control and illegitimately obtain data.

Assuming that clients and service providers interacting in a SOA have no interest in revealing their interaction to unrelated entities, the only way these entities can learn that an access has occurred is by using a covert channel, or in other words some information generated by the regular system's operation. There are two approaches to deal with covert channels: either avoid them at design time or reduce the channel quality at run time to avoid practical information leakage. Introducing noise or reducing the channel throughput

are ways to reduce the quality of an unavoidable covert channel.

One covert channel for personal data widely studied is the source and destination IP addresses of packets in a TCP/IP network. Even though the SOA specification does not require a TCP/IP network, most existing computer systems use this type of network and we can expect that various SOA implementations – including Web Services, its most popular realization – will use this technology for communication. Source and destination IP addresses were originally included in packets as a mechanism to route them through the nodes of a network and they were never intended to provide any information about the parties communicating.

However, a malicious observer able to watch all packets on a network can obtain important personal information, such as which hosts were accessed by a user (and therefore the services he used), or his location, only through analysis of source and destination IP addresses. Since this covert channel cannot be avoided, researchers developed techniques to reduce its bandwidth, i. e., reduce the amount of information flowing through the channel so that its practical use becomes unfeasible.

Early privacy-enhanced technologies proposed to reduce information disclosed by TCP/IP packet headers were created for e-mail applications. Solutions such as Babel [GT96] or the Mixmaster[1] follow the Mix design proposed by Chaum [Cha81]. Instead of sending a message directly to its destination, mail is redirected to an intermediary server, called a **Mix**, which forwards these messages to the destination using the Mix IP address as the source. Messages going from the user to the Mix do not contain the destination address on the IP header, and messages going from the Mix to the destination do not contain the original source IP on the IP header. To avoid correlation between messages entering and leaving the Mix, a number of messages from different sources is accumulated before the Mix forwards them to their destinations. This strategy creates a high latency when sending messages, but this is not a problem for e-mail traffic. To protect against attacks that compare the size of messages entering and leaving the Mix, all messages are padded to a fixed size.

Figure 2.4 depicts the operation of a Mix. Users A, B and C send messages (represented by envelopes) with different sizes at different moments to the Mix (represented by $t = 1, 2, 3$). Under each envelope are the source and destination addresses of the packet containing the message. The message is addressed to the Mix and the final address is inside the packet instead of in the TCP/IP header. The Mix waits for a number of messages before simultaneously forwarding them to their destinations ($t = 4$) with a fixed size. This design reduces the IP covert channel bandwidth, but requires users to absolutely trust the Mix; a malicious Mix can easily correlate source and destination IP addresses of messages. Further versions implement chains of Mixes to distribute trust among different servers: instead of passing by a single Mix, each message traverses a chain of Mixes in such a way that only the first Mix knows the address of the user who sent the message and only the last Mix knows the message destination. This modification of the protocol can resist even to the collision of some Mixes to reveal the source and destination IP addresses of a message.

[1] mixmaster.sourceforge.net

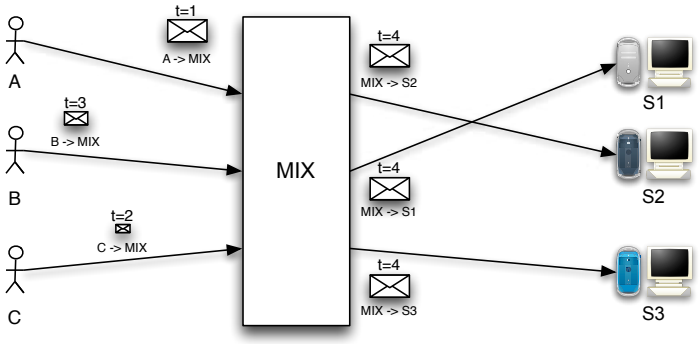


Figure 2.4: Basic Operation of a Mix

However, the problem of how to route message replies still remains. In the above design, since a message does not contain its original source address, the destination cannot reply to the message. Early solutions required the creation of long-lived pseudonyms on the Mix server, but this is undesirable since an attack against the Mix can correlate a user with a pseudonym and discover all messages sent to him. Babel introduced a mechanism that allows the source of a message to add to the message a return path (Return Path Information - RPI). The RPI is created in such a way that only the final Mix of the return path (the one closest to the original source) knows which address started the communication. Mixminion [DDM03] perfected Babel's mechanism for message reply to avoid some attacks that could reveal information about a message's source address.

The same approach, however, could not be used to reduce the bandwidth of the IP address covert channel for Web applications. Web traffic is interactive and requires less latency than e-mail, so the original Mix design is not appropriate. The Anonymizer[2] is one of the earliest approaches to provide anonymity in Web surfing and is one of the few privacy-enhancing products on the market. In the beginning it consisted of a proxy server that received the user query, removed any personal identifiable information and forwarded the query to the destination, similar to a mail Mixer, with all queries appearing to the destination as coming from the Anonymizer proxy. To reduce latency, however, the original Anonymizer did not accumulate messages before forwarding them, as Mixers do. This approach for Web privacy relies on the proxy to provide anonymity; if the proxy is attacked or if it is malicious, anonymity is lost.

Web MIXes adapted the Mix approach to the Web requirements [BFK01]. A local proxy (the Java Anon Proxy – JAP) is installed on the user machine, and is responsible for stripping user identifiable information from Web requests and to forward them to a chain of Mixes. To provide anonymity without increasing latency, the JAP of each client periodically generates dummy traffic to Mixes. This traffic is mixed to client requests in such a way that each request is mixed to the same number of packets from other users (real requests or dummy traffic).

[2]www.anonymizer.com

Crowds [RR98] relaxes some of the requirements of Mixes and proposes an approach inspired by the behavior of humans in the real world, that blend into a crowd to become anonymous. In Crowds, each user runs a process that works as a local proxy for Web requests, and form a group with other users running that process (the "crowd"). Whenever there is a new request, the local proxy randomly decides if it is going to forward the query to its destination or to another member of the crowd. When a member of the crowd receives a message, it also randomly decides if it forwards the request to its destination or to another crowd member. As a result, the request will eventually reach its destination, but its source can be any member of the crowd. This avoids correlation between the source IP address of a Web request and the user originating it. Any member of a path, however, has still access to the destination IP address of the request.

Even though the approaches above were relatively successful, they are still application-specific (e-mail and Web traffic only). Any application that uses TCP/IP networks are vulnerable to the IP source and destination covert channel, so a more general approach to reduce the bandwidth of this channel is required for other applications. Onion Routing [GRS99] was designed to support multiple protocols in addition to HTTP (Web) and SMTP (e-mail). It is composed of three software layers: one application-specific filter that sanitizes application data to remove personal information, an application-specific proxy that translates the packets to an application-independent format, and the onion proxy that builds and manages the chain of Mixes. However, to keep the latency of the chain of Mixes as low as possible, packets in Onion Routing are forwarded as soon as they reach a Mix which means that, if the network load is low, an attacker observing the packet flow through the chain of Mixes may be able to relate the source IP of a packet entering the chain with the destination IP of the corresponding packet leaving the chain [RSG98].

The original design of the Onion Routing network had many flaws, and as a consequence the network was unstable and contained only a few nodes. Tor [DMS04], the second generation Onion Router, fixed many of these flaws and extended the architecture to support, for instance, the distribution of node state information using directory servers or the support for hidden services, which are services that advertise some onion routers as their entry points (or *rendezvous* points) for clients, keeping the real service address secret. These improvements led to a much more stable network that provides an acceptable service level for regular Web access. Pre-compiled packages such as Vidalia[3] simplified the installation of the software required to be a Tor client or a Tor relay (a Mixer), which also contributed to increase the network size.

Other approaches were inspired by Onion Routing but instead of an architecture where few nodes are responsible for mixing requests and forwarding them to the destination they propose a peer-to-peer approach, where all nodes are Mixes such as in MorphMix [RP02]. The appealing reason for a peer-to-peer architecture is that it provides a bigger throughput and complicates traffic analysis. Tarzan [FM02] goes a step further in protecting against the IP address covert channel by also introducing noise on the channel by means of cover traffic, aimed at hiding the communication patterns of the nodes. However, noise generation and transmission is costly in terms of network usage (and consequently in terms

[3] vidalia-project.net

of energy for resource-constrained devices), so this approach should be used only when a global eavesdropper (one that is able to watch all packets being transmitted on the network) is a relevant threat.

An original approach to Mixes is proposed by pMIX [MD05], that combines a Mix to a Private Information Retrieval protocol (PIR, discussed in Section 2.2.3) to enable anonymous communication between two entities using a single Mix in such a way that even the Mix administrator cannot define which parties are communicating. The source of a communication requires a slot from the Mix and writes data to that slot. Destinations request data from a slot to the Mix using a PIR protocol, which prevents the Mix to identify which slot the destination is requesting. This idea, however, has a number of drawbacks when considered for the mobile scenario: parties communicating must privately exchange their slot numbers before communication, cover traffic is used to hide communication patterns which creates a communication overhead and the Mix has a large computational overhead due to the PIR protocol. Further work reduced this computational overhead [MD06] enabling support for anonymous communication in situations where few communications are simultaneous.

The extension that reduces the computational complexity of pMIX uses another form of anonymous communication called a DC-Net, based on the Dining Cryptographers Problem [Cha88], which serves as the base for another series of protocols for hiding the source and destination IP addresses of a communication. This problem was originally presented in the following manner: three cryptographers are dining at a fancy restaurant when the waiter tells them that their dinner was already paid for. The cryptographers want to know if their company paid for the dinner or if it was one of them, without revealing the identity of the generous cryptographer. They propose thus a protocol for anonymously sending messages to all dinner participants.

Each cryptographer shares a key with all other cryptographers, and each one broadcasts the XOR of all the keys he possesses (if he did not pay for dinner) or the opposite result (if he paid for dinner). After receiving all broadcasts, each participant computes the XOR of the packets received: if the result is 1, a cryptographer paid for dinner, if it is 0, their company paid. Since all participants broadcast their results and every broadcast is required to recover the final message, the identity of who paid for the dinner remains a secret. This simple protocol can be extended for situations where more than one participant tries to transmit a message at the same time (collision handling) and where a message is targeted to a single participant (using cryptographic keys). Figure 2.5 shows how the protocol works. Each participant shares a key with the others, and broadcasts the XOR of the keys he possesses; participant A, who paid for the dinner, broadcasts the opposite. Every participant computes the final result as the XOR of the values broadcasted and as the result is 1 ($0 \oplus 0 \oplus 1$) each participant knows that someone paid for dinner, but cannot tell that it was participant A.

This approach provides unconditional security, which is a stronger notion than computational security provided by the protocols based on Mixes and public key encryption. Computational security means that the protection the algorithm provides depends on how difficult it is for a resource-bounded adversary to solve a given difficult problem

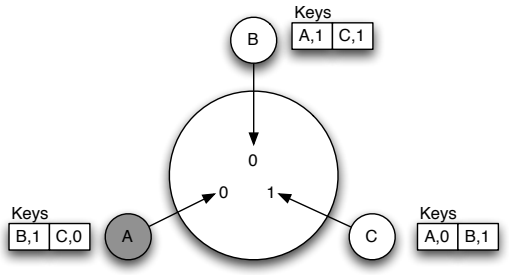


Figure 2.5: Communication on a DC-Net

(such as integer factorization in RSA [RSA78] or the discrete logarithm problem in Diffie-Hellman [DH76]); if an efficient method for factorization is discovered, for instance, the protection provided by RSA disappears. Unconditional security cryptographic primitives, on the other hand, offer protection even against resource-unbounded adversaries, which means that if an adversary has access to substantial processing power and has a lot of available time, it is still unable to breach the protection. DC-Net privacy protection is based on the logical XOR function, which is not reversible. An adversary that has access to the outcome of one round of the protocol can use infinite resources and still will not be able to discover who sent the message.

Although very elegant, this solution has a series of drawbacks. First of all, the communication overhead imposed by the protocol is unacceptable in pervasive environments. The use of broadcast for normal communication is very inefficient both in terms of network usage and energy consumption. Even if multicast is used, a network with N members would require the transmission of N bits of traffic for 1 bit of message, causing an overhead of $N - 1$ bits. Second, the protocol requires that every node share a key with every other participant, which can be too costly in ad hoc environments that cannot rely on a Public Key Infrastructure. Finally, collision handling requires more rounds of the protocol, which increases even more its performance overhead.

Some approaches try to solve these issues to provide a feasible implementation of DC-Nets that reduces the IP covert channel throughput. Herbivore [GRPS03] organizes network nodes into smaller cliques to reduce the performance impact of using broadcasts and the management cost of distributing keys, and also provide a bandwidth reservation mechanism to avoid collisions. Each clique contains at least k members, where k is the anonymity level required for the network, and nodes only need to share keys with the members of the clique. P^5 [SBS02] (which stands for Peer-to-Peer Personal Privacy Protocol) is another protocol based on the DC-Net approach. Instead of organizing nodes in cliques, P^5 creates a hierarchy of broadcast channels to limit the performance impact of broadcasting packets. User groups are organized in a tree and their position affects how messages are broadcasted: messages sent to a group closer to the leaves generate less broadcasts than messages sent to groups closer to the root. However, reduced broadcasts impact anonymity, and groups down in the hierarchy offer less anonymity than groups

up on the hierarchy. Users can choose the group to which they belong according to their performance and anonymity requirements. P^5 also uses dummy packets to generate noise and complicate traffic analysis. Periodically, every node that is part of P^5 randomly selects a broadcast channel to which it sends a noise packet. This noise can be a packet already transmitted by the node to some other channel, a packet locally generated by the node, or a noise locally generated by the node. An external observer cannot tell the difference between any of these packets and a real transmission packet.

The IP covert channel is just one of the many covert channels that exist in pervasive computing and that can be used to invade the privacy of individuals. Covert channels are hard to detect and the identification of all covert channels in pervasive computing is an open issue. Techniques for detecting covert channels were proposed [McH95], but they fail to detect channels that may appear due to unexpected circumstances, such as the activity of LED status indicators [LU02]. Additionally, strategies to protect against those channels can introduce new covert channels themselves [Mur07]. For instance, by counting the packets transmitted by each node in Tor, one can track communications and infer sources and destinations [SS05]. More research is needed to find further covert channels in service-oriented pervasive computing and to create strategies to avoid that unrelated parties obtain information about the existence of a service access.

2.2.2 Protect the Content of an Access

The problem of protecting the content of a service access is the same as avoiding leakage of data on a *legitimate* communication channel. Legitimate channels were defined by [Lam73] as regular communication channels dedicated to information exchange. The solution to this problem is to ensure the confidentiality of the communication channel, in such a way that only the parties involved in the communication are able to understand its content. Confidentiality is usually obtained through cryptography using symmetric and asymmetric encryption algorithms.

Onion Router and Tor, presented in Section 2.2.1, use public key cryptography to protect user data traversing Mixes. In the original Onion Router, data is recursively encrypted using the public key of each hop creating multiple layers of encrypted data (forming the "onion" of the protocol's name). Each layer of data contains next hop information in such a way that, when an intermediary router receives a packet, it is able only to decrypt one encryption layer and access the next hop packet information. Intermediary routes do not know neither the packet destination nor its content, which is encrypted using the public key of the destination.

This approach has two main drawbacks. The first is that public key encryption is process-consuming, and when it is used to encrypt every packet that is part of a communication, it introduces a noticeable delay that may discourage use of the protocol. The second problem is that this strategy does not guarantee perfect forward secrecy. Since packet content is protected using a long-lived public key, an attacker can log all user communication, force intermediary routers to release their private key and have access to the communication contents. Even though this situation seems unlikely, this is exactly what

happens, for instance, when a government wants to investigate the activities of a dissident that transmitted messages using the protocol.

Tor improved the original Onion Routing protocol to reduce its overhead and obtain perfect forward secrecy. Clients negotiate a session key with each router on the path between the client and the destination using the Diffie-Hellman key agreement protocol [Sch96]. To certify that the key will be established with the correct router, the client sends the first part of the key agreement encrypted with the router's public key. This protocol allows for the creation of a one-side authenticated session key: the client authenticates the router but the router cannot authenticate the client. The protocol is executed only once when the connexion is initiated; further packets that make part of the connection are encrypted using the symmetric session key. This approach brings two benefits: first, symmetric encryption is less computationally expensive than asymmetric (public key) encryption, which reduces processor usage and the protocol delay, and second, the session key can be erased after the communication is concluded, which provides the property of perfect forward secrecy since the key used to encrypt a session cannot be recovered after its end.

Both of the approaches presented above rely on a Public Key Infrastructure (PKI) to ensure the confidentiality of communication between a client and a service provider. A PKI provides the required infrastructure for the generation, revocation and distribution of digital certificates, which attest that a given entity is associated to a specific public key. PKIs rely on a certifying authority (CA) to ensure that the certificate is authentic [Koh78]. The CA is a critical element of the architecture since it must be deeply protected and must not be controlled by an attacker (to avoid generation of fake certificates). This requirement is too strong for ad hoc networks that cannot rely on an infrastructure. Some works have been proposed to adapt existing PKIs to ad hoc networks, for instance by intelligently replicating the CA in such a way that the PKI can resist to the compromise of up to t replicas, where t is a security parameter [ZH99], but certificate revocation remains an issue.

Since PKIs may not be suitable to situations where clients and service providers do not have access to an infrastructure, alternative solutions for protecting the content of an access which are not based on public key encryption must be investigated. In Chapter 3 we propose a protocol that explores the heterogeneity of HMANETs (as described in Section 1.2.4) to protect the content of an access using only symmetric cryptography. Keys are exchanged according to a protocol that resists to attacks with certain characteristics. After exchanging keys, nodes communicate using messages encrypted with the symmetric key previously exchanged.

2.2.3 Limit Data Disclosed for an Access

During a service access, users and service providers disclose relevant information, including contextual data, to obtain a mutually agreed outcome. In pervasive computing systems, context plays a key role to allow for better service customization as it can help applications to proactively anticipate user needs minimizing their intrusiveness [Sat01]. However, con-

textual information is rich and data used for system customization can indirectly reveal critical personal data. To avoid such situations, information disclosed during a service interaction must be reduced to the minimum required to complete the transaction.

Data can be represented in different resolutions. For instance, location can be expressed in terms of geographic coordinates with smaller accuracy, e.g., a country name, or with bigger accuracy, e.g., GPS coordinates. To limit data disclosure, protocols must be designed to require the least amount of personal information from the entities participating in an interaction and, whenever relevant, they should support personal data with multiple levels of granularity to enable each entity to define its exposure to a privacy invasion.

When limiting data disclosure, individuals can use 3 techniques: multiplication, generalization and modification. More formally, consider I as the set of all possible values for a personal information such as age or location, and $i \in I$ as an instance of that information (such as "32" or "1600 Pennsylvania Avenue"). Consider also G as the set of another type of personal information and $g \in G$ one element in the set. We define:

- **Modification:** A function $f(x) : I \rightarrow I, f(i) = i'$ and $i \neq i'$
- **Multiplication:** A function $f(x) : I \rightarrow I \times ... \times I, f(i) = \{i_1, ..., i_n\}$ and $i \in \{i_1, ..., i_n\}$
- **Generalization:** A non-injective function $f(x) : I \rightarrow G, \forall f(i) = g, \exists f(i') = g$ such that $i \neq i', i' \in I$

In other words, modification substitutes the requested information for a different information from the same set (semantically equivalent). Multiplication replaces a specific data with a set of n data where the requested data is contained. Generalization substitutes the information requested by an information from another set that is related to multiple data from the original set. It is important to notice that only in the multiplication approach the real information remains part of the resulting set. In both modification and generalization the original information is lost in favor of a different or less specific information, the main difference being that modification maintains the semantics of the original data (the result is a different element from the same set) while generalization changes the semantics (result is an element from a different set).

Contextual and personal information may also be expressed as a combination of heterogeneous data. For instance, the identity of a person can be expressed as the full name plus the date of birth. The techniques defined above can be independently used on each data, providing different levels of resolution to the composed information.

One of the pioneer efforts to enable users to reduce the amount of personal information disclosed during a service access was the Platform for Privacy Preferences, or P3P [RC99], proposed by the World Wide Web Consortium (W3C). P3P is a protocol that allows users to define their policies concerning personal data disclosure and service providers to define their privacy policies on a format that can be programatically compared, without user intervention. P3P comprises also a protocol for negotiation when the user's preferences and the provider's policy do not match, and a language for the specification of user-side preferences concerning disclosure of personal information [CLM02]. Based on those preferences

users can define which data they are willing to disclose and service providers can specify which data is required for service access. Even though P3P was designed for the Web environment, the ideas proposed could be as well adapted for service-oriented pervasive computing, particularly when using Web services. The Privacy Awareness System (*pawS*), for instance, adapted a number of ideas from P3P and introduced some new features to propose an architecture for the negotiation of privacy preferences in ubiquitous computing systems [Lan02].

Location is a particularly sensitive contextual information, and for that reason many research efforts focus on how to modify location data resolution to reduce the possibilities for privacy invasion. One such effort considers location as a tuple $< period, region >$ and reduce the information resolution by modifying information about the period and region to include a higher number of individuals [GG03]. Another considers location as a tuple $< id, location >$ and modify the identification data to associate location data from the same user to different pseudonyms [BS03].

The generalization technique applied to location data explores the fact that location data can be organized geographically to represent different levels of precision. Mist [AMCK$^+$02], for instance, organizes location data hierarchically according to precision (e.g., building $\rightarrow$ floor $\rightarrow$ room) and creates an overlay network of routers that clients can use to generalize user location. More general routers offer better privacy but increase routing overhead and decrease performance.

Multiplication was also suggested as a way to reduce location precision. One proposed approach finds other individuals on the same region and supply the location of all members of the group instead of only the specific individual location [DK05]. Consecutive observations of the group however could reveal the individual identity by computing the intersection between the groups. An orthogonal approach is to simulate location data instead of using data from other users, allowing for provision of the real location along with dummy data to reduce the effects of information disclosure [KYS05]. However, also in this case continuous observation of location data may allow inference of which data is real and which is fake.

Images are another type of information that raise privacy invasion concerns. Few initiatives use the modification approach to control image data resolution, for instance changing the captured video to only display "socially correct" images [CCB00]. The greatest part generalizes the video image using a myriad of image filters to reduce its resolution [HS96, BEG00, ZS98, NG03]. Such approaches have limited efficiency since an individual may be recognized by his peers not only from his features but also from other properties harder to dissimulate through a filter such as the way he dresses or the way he moves.

Identity management is also a major concern for privacy protection. A large number of applications require some form of client authentication, but systems usually follow bad practices from the real world – e.g., showing an ID to prove your age – and require from clients more information than necessary to perform a transaction. Group signatures [CH91] and more recently ring signatures [RST01] were proposed to securely relate a message to a group of users instead of a single user, such that the user who originated the message

remains anonymous in relation to the group, generalizing his identity. Brands [Bra00] created techniques to increase privacy of digital certificates, including ways to disclose a single property of the certificate or proving a proposition about a property without disclosing it, also generalizing user identity to all users featuring the same property. The PRIME Project [Con05] aims to implement these techniques in the real world. Modification and multiplication are probably not applicable to authentication since they would require creation of forged identities, which is exactly what authentication tries to avoid.

Finally, database research proposes approaches that try to increase privacy of information stored in a database. Statistical databases [AW89], for instance, were proposed as a form to provide clients with some information about records in a database, without revealing any specific record. This is usually achieved by generalizing the contents of the database through a statistic of the values stored such as the mean, standard deviance, or number of records. These statistics, however, can still reveal too much information about the database depending on the data distribution, so this technique is usually combined with modification to increase privacy protection. One such approach is perturbation [TYW84], where the database system introduces noise in the values of records and then a statistic is calculated that provides an estimator to the real metric. Another approach is to select a random sample of the database and replace the correct metric regarding all records to the corresponding metric regarding the random sample [Den80]. It has been proven, however, that any approach providing statistically correct answers can be compromised [Bec80]. The main drawback of such approaches is that users must trust the database to provide privacy-enhanced results to queries.

Another related subject in database research is how an individual can use contextual and personal information to query a database without revealing data from the query to the database server. The trivial solution is to allow a user to download the whole database and perform the query locally, but there may be more efficient alternatives in terms of communication. This problem is called Private Information Retrieval (PIR) [CGKS95], and many solutions were proposed for the centralized case where the query is performed on a single database with bounded computational power or for the distributed case where the query can be performed on k non-colluding copies of the same database. The survey by Gasarch [Gas04] summarizes existing results, but the basic idea is to generalize the query through encryption to receive a larger set of encrypted records in such a way that only relevant records for the query submitted can be accessed by the user. A similar problem is how to protect the database from the user, in such a way that his knowledge of the database is also limited [GIKM98]. Although early results showed that PIR is impractical when implemented in available hardware [SC07], recent developments in hardware processors allowed creation of the first usable PIR schemes [WS08].

2.2.4 Control Usage of Data Disclosed for an Access

The final challenge for increasing privacy during service access is how to ensure that private information disclosed for the access purpose will not be used to other purposes by the entities participating on the interaction. When a user requests directions from a

navigation system, how can he be sure that this information is not shared with his car assurance company or with a marketing agency? Since digital information can be easily copied and distributed without losing resolution, users should be able to control how their private information can be used and disallow secondary uses that go against their interests.

Usage control requires (i) languages and protocols to define data usage rules and preferences and (ii) tools and mechanisms to enforce compliance to specified rules and preferences. Concerning (i), the most relevant initiative is the Platform for Privacy Preferences, or P3P [RC99], already discussed in Section 2.2.3. P3P allows users to define how they want their data to be used, and service providers to define how they want to use the data (e.g., the data retention policy or if data is anonymized [Cra02]). PawS [Lan02] extends P3P for ubiquitous computing applications. The main criticism about these protocols is that enforcement of usage rules depends on the cooperation of data consumers, and this is a requirement as reasonable as believing that users will not distribute songs, if the same reasoning is applied to the copyright scenario [Cat00]. When data consumers are not willing to cooperate, P3P does not provide an enforcement mechanism that guarantees that usage rules will be respected.

Requirement (ii) above, thus, is the most challenging technical aspect of usage control. Hippocratic databases [AKSX02] attack the problem of enforcing usage rules for records in a database. A hippocratic database is a re-designed database management system (DBMS) that considers privacy as a fundamental property. It provides features to protect information usage such as storing the purpose specification along with the data, limiting data retention and requiring user consent before collecting data. It must also enable users to verify if the database adheres to the usage rules they define. Published results [ABF+04] propose an auditing mechanism that allows users to verify if the database controls access to their personal information according to their preferences, but it requires cooperation from the server and thus the database server must be trusted.

The problem of enforcing usage rules on disclosed data bears some similarities with the problem of controlling usage of digital copyrighted works. In both cases an entity that has some rights over a certain data wants to allow some uses of that data while limiting other uses. In the copyright scenario, Digital Rights Management (DRM) is presented as a mechanism to enable content providers to control the manipulation of copyrighted digital material by content consumers. Based on the DRM approach, researchers proposed the creation of a Privacy Rights Management system (PRM) [KK03]. It considers individuals as content providers that can define usage rules for their content, which is their personal information.

The main problem with this idea is that DRM is designed for the specific scenario of publishing, where a small number of publishers provide a relatively small number of copyrighted works to many consumers, while in PRM a huge number of individuals provide vast amounts of private data to a few service providers. The effect of this reorganization of forces in the architecture scalability is unknown, and it remains to be seen if the DRM architecture scales when applied to privacy protection. Besides that, the adversary model in the privacy scenario is different from the adversary model in the copyright scenario. DRM technologies are designed to protect copyrighted material against the average computer

user, who has limited knowledge about computer internals and that has access to a very short amount of resources. In the privacy scenario, however, the typical adversary are big companies or governmental agencies that are the exact opposite of the DRM adversary: they have advanced computing skills and access to powerful computing resources. Against such adversaries, DRM protection techniques are not very efficient.

Others also proposed architectures for controlling and enforcing data usage. Confab [HL04], for instance, follows the principle that user data should be processed at the user's device as much as possible. It is based on a privacy-aware tuple space that allows users to specify meta-data associated to each tuple containing user personal information. The meta-data specifies privacy preferences that apply to that tuple, such as when the tuple should be deleted and how many times the tuple can be accessed. Every access to the tuple space is mediated by Confab, which ensures that tuples usage rules are respected.

Finally, [PHB06] proposes a server-side architecture for usage control. The user specifies a usage control policy containing provisions, which are access control requirements, and obligations, which are future usage control requirements. Obligations are then divided into controllable, those that can be enforced, and observable, those that can be monitored to identify a violation. If an observable obligation is violated, the user can search for a compensation from the violating party. Whenever a consumer requires some data, the server sends first a contract containing the data usage policy which contains the provisions, obligations and compensations the data owner requires to allow his data to be used. If the consumer accepts the contract, the server sends the requested data. Enforcement of controllable obligations is not detailed, but the authors also suggest the use of DRM-like mechanisms. Then again, the user must trust that the server will respect the specified usage control policy.

The trust model that we defined in Section 1.2.4 assumes that devices in a pervasive computing environment cannot be trusted. Indeed, pervasive computing environments are ad hoc, so users do not necessarily know each other in advance. They are also open so users join the environment with their own devices and are responsible for their administration. In this setting, the approaches presented above are not suitable. All of them require cooperation of data consumers, and this requirement is not realistic in the environments we are focusing on.

2.3 Privacy Research in Service Discovery

There are two aspects of privacy in service discovery. The first is when we consider the middleware-provided service discovery feature as a service by itself. In that case, service discovery requests are no different than service accesses, and most of the technologies discussed in Section 2.2 can also be applied to improve the privacy of entities using service discovery, for instance to ensure that the content of service announcements are only disclosed to entities participating in the protocol.

The second aspect is when we analyze the service discovery directory as a third party participating on a service access. In this case we must evaluate how intruders can use

information from the service discovery protocol to obtain data about the access. Indeed, information disclosed in service descriptions used for announcements and requests contains data about a service that will eventually be consumed later. Even if the access to the directory is private, data used in service discovery – e.g., service descriptions or contextual data – can still reveal information about the access. In the next sections we discuss how to protect service access from the four privacy attack strategies described in Section 2.1.1 when they are used with service discovery data.

2.3.1 Protect the Existence of an Access

Before accessing a service, clients need a reference to the service and the service provider. This reference may be statically defined on the client side, but SOAs provide a service discovery feature that helps to reduce coupling between clients and services. Service discovery enables clients to dynamically discover available service references and more easily adapt to environment variations.

In general, a service discovery request precedes a service access. As a consequence, such requests reveal information about the existence of a service access that will be performed later. Surely, the service access corresponding to a service discovery may never happen, but data exchanged during discovery gives an indication about the access that follows. Analysis of consecutive service discovery requests can easily reveal if the access has occurred or not. Service announcements also contain service-related information that a service provider may want to disclose only to entities allowed to access the service.

Zhu *et al.* [ZMN05] identified two types of service discovery protocols concerning their required infrastructure they require: directory-based and nondirectory-based. In the nondirectory-based model, queries and announcements are sent to all entities listening to a specific channel (through broadcast or multicast), and all entities sharing the same channel can identify that a client is looking for a service. This impact on privacy results from the protocol architecture, since lack of a service directory forces advertisements and requests to be revealed to all users.

When considering the directory-based model, the service directory concentrates descriptions contained in service provider announcements and client requests. This reduces disclosure of description data to a smaller number of entities; those entities however gather data on all services announced and being discovered inside a domain. The service directory, thus, is critical for protecting the existence of a service access.

PrudentExposure [ZMN04] is a service discovery protocol that protects the existence of a service access by reducing disclosure of the content of a discovery request only to domains for which the client possesses credentials. Service directories from domains that the client cannot access do not receive the request. To prevent service directories from learning all the credentials possessed by a client, domain information is disclosed obfuscated by a Bloom Filter [Blo70]. Directories verify if the client is member of one of the domains they manage, and send back another Bloom Filter to confirm. Only after this confirmation is that clients authenticate with relevant directories and send the service request.

Directories in PrudentExposure are trusted entities as they are responsible for authen-

tication and have access to all information regarding service discovery inside a domain. Besides, in this approach clients still have to authenticate with directories that may not contain the requested service since authentication is performed after domain match and at that moment the client still does not know if the repository contains relevant services, thus unnecessarily disclosing personal data from service requests.

To reduce service request disclosure only to directories possessing relevant service descriptions, the same authors proposed Progressive Exposure [ZZMN05]. In this protocol, a service discovery consists of multiple rounds of message exchange. On each round the client generates a code-word that reveals partial information about service request and domain membership. If the client or the service provider forecasts a negative outcome on any round, they can promptly abandon the protocol before disclosing complete service and domain information; clients only authenticate with directories that have the description of the sought service and that manages domains where the client is a member. Nevertheless, code-words are generated from a hash of the service description, and the consequence is that only exact syntactic matches between requests and announcements can be found.

2.3.2 Protect the Content of an Access

Some data that are part of a service specification can be later used during a service access. For instance, when a user looks for a tennis court reservation service in his current city, he may have to reveal his location again when accessing the service. Even though private data can be protected during service access using the techniques from Section 2.2.2, it must also be protected during service discovery to avoid privacy invasions.

Clearly, the same techniques used to avoid that intruders access data exchanged between clients and service providers can be used to protect the communication between a client and the service directory or a service provider and the directory. SDS [CZH+99], for instance, uses asymmetric encryption to protect data sent from clients and service providers to service directories. In SDS, clients, SDS servers and service providers possess public keys and certificates that can be used for authentication. All information exchanged between a client and an SDS server or a provider and an SDS server are encrypted and thus protected from eavesdroppers. As a result, data that a client discloses when accessing the service discovery feature cannot be obtained by intruders.

However, there is also the problem of how to protect private data from the service directory itself. Considering SDS, data is protected during transmission, but SDS servers have complete access to the data. This may be undesirable for clients and service providers since directories are not relevant for a transaction and should not obtain details about an access. At the same time, directories are responsible for finding services on behalf of clients and some details about the access are necessary to allow the directory to perform the task.

Is it possible to solve this conflict? To the best of our knowledge no service discovery protocol considered this issue. All solutions assume that the service directory is trusted and that personal information disclosed to the directory is protected and will not cause an impact to the privacy of clients and service providers. Unfortunately, there are a number of situations where service directories must be highly available and distributed, which makes

it harder to comply to the requirement of trusted directories. In Chapter 4 we propose a service discovery protocol that protects the information that clients disclose in service descriptions, and that allows service directories to find relevant service descriptions using these protected service descriptions.

2.3.3 Limit Data Disclosed for an Access

As we explained in the section above, some of the data disclosed in a service description may also be used during a service access. For instance, users may disclose their location during service discovery to find services nearby, and disclose the same location during service access for service customization. Clients and service providers not only need to protect this data from entities unrelated to the service access, but must also be able to disclose the least amount of information that still enables them to find and publish services as a way to reduce the impact on privacy of disclosing personal data.

Progressive Exposure, discussed in Section 2.3.1, also helps to limit data disclosed for an access enabling clients to progressively disclose service descriptions while checking the probability that the service directory contains the description that the client is looking for. After each round of the protocol, the client knows that either the probability that the directory contains the service increases or the directory does not contain the service. This allows a client to avoid disclosure of sensitive personal data to directories that are not relevant to the service discovery being performed. However, a directory that possesses a description matching the client's query still has access to all data disclosed in the description.

Semantic service discovery protocols [PKPS02, TBG01, Ben07] also help users to control the granularity of the information disclosed during service discovery. Instead of specifying services and properties syntactically, in semantic service discovery protocols clients and service providers can annotate a service description with concepts belonging to an ontology. The ontology maps the relation between concepts in a domain, which allows the service directory to find not only service descriptions that match exactly the one that the client requests, but also other descriptions annotated with concepts related to the concepts the client specified. As a result, clients may specify services using less precise concepts (e.g., a *shop* instead of a *flower shop*) and locally filter out the results that do not match the more specific description.

2.3.4 Control Usage of Data Disclosed for an Access

In the sections above we discussed the privacy effects on a service access of data disclosed during service discovery and how this data can reveal the existence of an access, the content of an access and reveal more data than necessary for an access. Those issues are complementary to the privacy risks of using the service discovery feature.

However, regarding data usage control, service discovery does not introduce any additional risk to privacy in service access. If users are able to define that data disclosed during

service discovery must only be used for service discovery, there is no risk that this information will later be used to correlate discovery data with service access information. Data usage control for service discovery suffices to reduce privacy for service access concerning service discovery data.

As discussed in Section 2.2.4, data usage control is mostly an open issue. Existing approaches require the cooperation of parties that access the data, and even though this requirement may be reasonable in some specific situations, it is not clear if usage control is at all possible in open environments such as pervasive computing, where unknown users and untrusted devices access protected data. Much more research is needed to answer these questions before a feasible solution for usage control is foreseeable.

2.4 Privacy Research in Service Composition

There are two basic paradigms for service composition [SH05]. In **orchestration**, a single entity is responsible for calling all services that are part of a composition and for obtaining the results of each call. On the other hand, **choreography** is a more distributed paradigm where services invoke and send results to each other, without a central entity. Even though choreography is more flexible because it does not depend on a central authority, it is harder to accomplish in open environments where entities do not know each other in advance and are mutually distrusted. Orchestration, thus, is more appropriate for pervasive computing.

Service composition can be considered as a service by itself as well. Clients specify service compositions and submit them to the middleware-provided service composition feature, which is responsible for evaluating if the composition can be executed with the services available in the current environment and also for proposing a set of services to perform user-defined tasks. Intruders should not be able to discover if a user required a service composition nor the tasks specified in the composition. Users must be able to disclose the smallest amount of personal data when specifying a composition and that data must be used only for realizing the composition.

Nevertheless, there is a second dimension to the privacy impact of a service composition mechanism which concerns its interaction with other middleware-provided features, namely, service discovery and service access. Service composition relies on service discovery and the interaction of those two features may disclose private information to intruders, since data provided for the first is used by the second.

A larger risk for user privacy appears from the combination of service composition and service access data. Information contained in a service composition defines how service access is going to be performed, and attackers can target service composition data to modify how clients access services in a way that a privacy attack is easier to perform. For instance, instead of providing a single service that requires various sensitive data, providers may offer simpler services that require different subsets of the sensitive data to make the user believe that the service is less sensitive than it really is. In the next sections we analyze the required privacy protection mechanisms for service composition and review existing research that can be used to implement such protections.

2.4.1 Protect the Existence of an Access

As opposed to service discovery, which involves a third party service directory, service composition is usually performed locally on the user's device. As a result, information specific to the composition is not disclosed to any additional entity since it is used on the client side. Realization of a composition involves a series of service discovery calls, which could reveal details of the composition to an untrusted service directory, so the problem of protecting the existence of an access in service composition can be reduced to the problem of protecting the existence of an access in service discovery.

If the service discovery protocol in place respects user privacy by implementing the protections described in Section 2.3, the service directory is unable to obtain personal information from service descriptions, and as a result cannot combine information from each request to infer service composition details. Service composition, thus, does not introduce any additional privacy issues concerning the existence of an access.

2.4.2 Protect the Content of an Access

The service composition defines which services the client will access from those available in the environment. Depending on the services defined to realize a composition, it can be easier or harder to protect the content of the corresponding access. The client may trust or distrust a service provider, may have or not its public key, and may be or not securely connected to the provider. These characteristics, among others, affect how easy it is for a client to protect the content of messages transmitted during a service access.

Chafle *et al.* [CCMN05] proposed to define constraints on a composition workflow that determine which entities are allowed to send and receive messages defined in the workflow. These constraints are specified as a set of rules that determine, for each message, which domains can originate the message and which can be its destination. Based on these rules, the system computes a partition of tasks into a distributed orchestration that respects the rules defined. Each part of the orchestration is executed inside a domain, reducing the need to exchange data among domains. This mechanism allows a client to specify constraints that simplify protection of sensitive messages.

Carminati *et al.* [CFH06] present a mechanism for secure Web service composition which allows service providers to specify their security capabilities (through SAML [OAS05c]) and security constraints (using WSDL [W3C01]). Security capabilities are the security-related functionalities that the service provides, and security constraints are the security-related capabilities that the service requires from other services. Based on these capabilities and constraints, a broker creates a composition that is compliant to the security requirements of service providers.

2.4.3 Limit Data Disclosed for an Access

When the client specifies a service composition, it must declare all data it is willing to provide to obtain a desired output. There is a conflict of interests at this point: the client

wants to maximize the possibilities of finding a suitable service composition and thus may be tempted to provide as much information as possible. However, disclosure of a great quantity of personal data can have a harmful effect on his privacy. How can a client find the correct balance?

There is an additional issue that is related to the problem of separation of duty [SS73]. To limit the privacy impact of disclosing personal information, users may want to ensure that private data that is sensible when analyzed together is processed by different service providers to avoid their correlation. Clients need mechanisms to avoid execution of service compositions that accumulate data in a way that is prejudicial to users.

Researchers proposed the definition of separation of duties for task execution on the user-defined workflow as a solution to this problem [BE01, KS01]. Their works assume that a set of roles are defined for each user executing tasks on a workflow, and ensure that no user can execute different tasks on the workflow requiring conflicting roles. These approaches support dynamic separation of duty, which means that the tasks of a workflow that a user can perform are not defined beforehand but depend on the tasks he previously performed on a workflow instance. For instance, a manager cannot approve a travel expenses report if he is the one submitting the report.

In pervasive computing our focus is on defining which service providers can execute which tasks on a workflow according to data they already accessed. The focus is less on tasks and more on data: a service provider can execute as many tasks as possible, as long as it does not access data that can be correlated to infer more information about the user. To adapt the above described approaches to service-oriented pervasive computing, roles should be dynamic since they depend on which data the provider has already accessed, thus increasing the complexity of the model. Definition of all roles that service providers can assume and how these roles can be associated to tasks would also be difficult.

Instead of specifying all possible partitions of tasks among service providers that respect the user-defined data isolation criteria, in Chapter 5 we propose a mechanism that enables users to reason about the privacy impact caused by the execution of a service composition. Based on this value, users can compare the impact caused by different executable workflows and choose the one that causes the smallest impact on his privacy. The privacy impact is determined based on the data accessed by each service provider and on the consequences on user privacy caused by the combination of this data.

2.4.4 Control Usage of Data Disclosed for an Access

If we assume that users are able to control how personal data disclosed during a service access can be used by service providers, and that users are also able to control how data disclosed during a service discovery can be used by a service directory, service composition introduces few additional requirements for usage control.

The specification of a service composition is used locally on the client side, so it does not require particular usage control measures. However, usage control rules defined for the composition must be carefully analyzed because they depend on the composition's task partition. At definition time, the user assumes a task partition and defines data usage rules

accordingly. Later on, the middleware proposes executable workflows that may not use the same task partition originally defined. The client then must reevaluate usage control rules to adapt them to the new task partition. This process may require another round of negotiations with candidate services if the user redefines data usage control rules.

Existing mechanisms for usage control enforcement such as [PHB06] or languages for description of usage control rules such as P3P [RC99] already support contract negotiation. Even though they are not suitable to pervasive computing (as discussed in Section 2.2.4) we can expect that if an effective mechanism for usage control in pervasive is ever created, it will feature a contract negotiation protocol. This protocol must be integrated into the middleware-provided service composition mechanism to ensure that usage control rules are still consistent with the executable workflows that the middleware provides.

2.5 Concluding remarks

As explained above, we believe that the problem of usage control in service-oriented pervasive computing is harder than generally assumed by existing research on this subject, and we do not think that an effective solution will be available in the near future. In the context of this thesis we decided not to attack this issue and we assume that clients cannot enforce how service providers use the data they disclose. As a result, the best strategy for the users to protect their privacy is to minimize data disclosure.

The middleware that we present in the following chapters includes mechanisms to increase privacy during service access, discovery and composition. According to our analysis of related work in privacy protection for service access, we conclude that more research is needed in mechanisms to protect the content of messages during a service access that are not based on public key cryptography. We explained in Section 2.2.2 why such mechanisms are important, and our contribution is described in Chapter 3.

With regards to service discovery, even though existing research proposes a certain number of solutions to improve privacy, they are not compatible with the availability requirements of service directories in dynamic environments such as pervasive computing, as discussed in Section 2.3. In Chapter 4 we propose a privacy-enhanced protocol for service discovery that reduces the trust requirements of service directories and simplifies their replication.

Finally, the impact of service composition on privacy is more deeply studied in the context of collaboration among companies, with well defined administrative domains and infrastructures, than in ad hoc environments. In particular, since we assume that usage control mechanisms are not available, it is even more important to understand how much private information service providers can discover based on data the user discloses. In Chapter 5 we present our contribution that enables users to reason about the privacy impact of executing a service composition, and to select among functionally equivalent compositions the one that has the smallest effect over their privacy.

I can't speak without an interception
This is private, please get off my line
Please tell me when I can have my privacy

The Kinks, "Party Line"

3

Multiple Paths for Communication Privacy in Service Access

The introduction of next generation mobile devices featuring multi-radio network connectivity enables the creation of hybrid mobile ad hoc environments as defined in Chapter 1. In such environments, which we call HMANETs, mobile nodes simultaneously or alternately use available network interfaces to augment the possibilities for user cooperation and data exchange. Multiple network connectivity enables users to offer services to a larger number of potential clients and also to find additional services that would not be available otherwise.

In HMANETs, user devices forward packets through heterogeneous network interfaces on behalf of other users. In addition to the network heterogeneity, HMANETs are also heterogeneous in terms of software platform. Android[1], OS X iPhone[2], Symbian[3], Black-

[1] developer.android.com/
[2] developer.apple.com/iphone
[3] www.symbian.com

berry OS[4] and Windows Mobile[5] are some of the operating systems installed on today's mobile devices. This heterogeneity can be beneficial for privacy protection since it makes harder for a single attacker to control the network.

In this chapter, we present a protocol for private communication that explores the network and node heterogeneity of HMANETs to increase privacy during service access. Our solution is based on three techniques: **multi-path routing** for discovery of multiple routes between the message source and destination, **adversary tolerable route selection** to select a set of routes that tolerates attacks with specific characteristics and a **private and probabilistically reliable message transmission** protocol that uses the selected set of routes to privately and reliably send messages between a client and a service provider. The main goal of our protocol is to divide a message into multiple shares and send each share to the destination through a different path. Distribution of message shares through heterogeneous paths require attackers to control multiple node platforms and network technologies, increasing the difficulty and costs of obtaining personal information in pervasive environments. Some of the ideas behind the design of this protocol were published in [CC08].

3.1 Challenges for Privacy Protection in Service Access

Privacy during service access is the issue that attracted most attention from the privacy research community. Relevant work in this area is reviewed in Section 2.2 and a number of interesting solutions exist today to avoid the four privacy invasion strategies described in Section 2.1.1. Those solutions, however, were designed with Web requirements in mind and some of them fall short to address the requirements of pervasive computing environments, particularly concerning the protection of the content of a message from eavesdroppers. When compared to the Internet, the lack of an infrastructure and the cooperative characteristics of pervasive computing raise new challenges for privacy protection in service access.

The first challenge is that, since pervasive environments typically incorporate ad hoc networks, messages are often routed by untrusted user devices. This is opposed to what happens with the Internet, where traffic is routed through dedicated routers. As a consequence, an attacker can volunteer to route packets on behalf of other users and benefit from this condition to analyze packets disseminated in the network. This analysis can identify ongoing service accesses and personal data transmitted in those accesses. Privacy protection mechanisms for service access on the Internet that assume trusted routers, thus, must be adapted to pervasive computing.

The second challenge is that pervasive computing applications may not have access to an infrastructure and cannot assume that a Private Key Interface (PKI) is available. This limitation complicates usage of public key certificates since without access to a PKI, users cannot verify certificate validity or obtain a Certificate Revocation List (CRL). A

[4] na.blackberry.com/eng/developers

[5] www.microsoft.com/windowsmobile

CRL defines a list of certificates that are no longer valid for reasons such as compromise of private keys or fraud during certificate generation. Without a CRL, users may believe that a certificate is authentic when it is actually compromised.

The third challenge is that public key encryption has an impact on the privacy of pervasive computing users. The public key associated to a digital certificate serves two purposes: it can be used to encrypt a message and ensure its confidentiality but it also automatically authenticates the message's source with the identity described by the digital certificate [Bra00]. As a consequence, every message is associated to a real world identity even when applications do not require authentication. It is important to notice that a number of daily activities do not require any form of authentication when performed without computers: people do not show their ID when checking the train timetable or asking for information. However, when such activities are executed with the help of computers, user identity can be unambiguously associated to each of these activities if public key encryption is used for protecting message content in pervasive computing environments. This situation is extremely undesirable since it leaves users with two options that are equally harmful to privacy: either encrypt service access and lose anonymity, or keep anonymity and expose service access content.

One approach to solve the above challenges is to distribute the entities that are part of a PKI to increase their reliability and distribute their trust [ZH99]. The certifying authority (CA), for instance, can be replicated in such a way that attacks to a certain number of replicas do not compromise the security of the PKI. This approach is more adapted to the requirements of an HMANET, but certificates are still associated to an identity. Brands [Bra00] proposed the creation of attribute certificates, which do not associate a public key to an identity but rather associate a public key to an attribute. Such certificates enable the use of asymmetric encryption without the need of authentication. Support to such certificates in infrastructure-less networks remains an open issue.

In this chapter, we present another approach to protect the content of a message in pervasive computing. Instead of using digital certificates to bootstrap a private communication between two nodes that do not know each other in advance (and thus have never exchanged keys), our protocol uses the multiple paths between the two nodes to distribute shares of a session key. As long as an attacker does not intercept a certain number of shares, the two nodes are the only nodes in the environment that are able to recompose the key from its shares. The protocol explores the network and software platform heterogeneity of HMANETs to minimize the risks that an adversary intercepts the communication, according to the adversary model defined in Section 3.2.

The first requirement to implement multi-path communication is that a node must be able to discover multiple routes to a destination. The node must also be able to send packets to that destination through different paths. In Section 3.3 we define a hybrid multi-path routing protocol that proactively keeps a single route to all nodes in the network but allows for reactive multi-path route discovery when necessary. It also uses source routing to enable the node generating a message to define the whole path that the message must traverse to reach its destination, allowing different packets for the same destination to follow different paths. Control messages from the routing protocol contain data that

identifies the network technology and software platform of HMANET components.

Based on the network topology that the routing protocol provides, nodes can select a set of routes that tolerate adversaries controlling networks or software platforms. This algorithm is defined in Section 3.4 and it first tries to define a set of paths that resists to an attacker controlling all nodes running a software platform and all networks with a given technology. If this is not possible, the algorithm tries to find one set of paths resisting to an attacker controlling all nodes running a software platform and another set of paths resisting to another attacker controlling all networks with a given technology. Since availability of such routes is dependent on the current network topology, it might be the case that neither of the outputs is possible. In this case, the protocol cannot guarantee any level of privacy.

Finally, after discovering the network topology and selecting a set of paths that tolerates attacks, the source node can use those paths to privately communicate with the destination. The protocol described in Section 3.5 uses the set of routes previously defined and a secret sharing scheme to encrypt the message with a key and then divide the key and the message into multiple shares. Each share is then transmitted through different paths in such a way that only the destination is able to recover the key from its shares and with high probability enough message shares reach the destination. After recovering the encryption key from the key shares and the encrypted message from its shares, the destination can use the key to decrypt the message and access its content.

In the next section we present the adversary model for HMANETs that our protocol for privacy-enhanced service access assumes. After defining and justifying the adversary model, we present the protocol and some experimental results. We finish this chapter with a discussion about the protocol evaluation.

3.2 Adversary Model for Privacy Protection in HMANETs

The protocol for privacy protection presented in this chapter tolerates attacks performed by a certain type of adversary. Before detailing how the attacks are tolerated, in this section we define an adversary model consistent with HMANETs and describe the types of attacks that can be performed by this adversary. The adversary model defined here serves as a base for the protocol description in the following sections.

An adversary model for network communication in multi-hop networks consists of four attributes: (i) which entities the adversary can control, (ii) how those entities behave under control of the adversary, (iii) how the adversary behaves during protocol execution and (iv) the computational power of the adversary. We say that an adversary controls a node if it is able to change the node behavior to act differently than defined by the communication protocol.

Regarding (i), researchers generally model a network as a graph where nodes represent devices and edges represent network connections. In this setting, the typical adversary model for controlled nodes is the **threshold** model: an adversary is allowed to control up to t nodes on a network with n nodes. This model, however, is inappropriate for many

real world situations because it assumes the worst case and overestimates the impact of an attacker. The network connecting nodes S and D in Figure 3.1, for instance, can resist to a threshold adversary with $t = 1$, since there is a set of two nodes (marked with letter A) that can interfere with the communication between S and D.

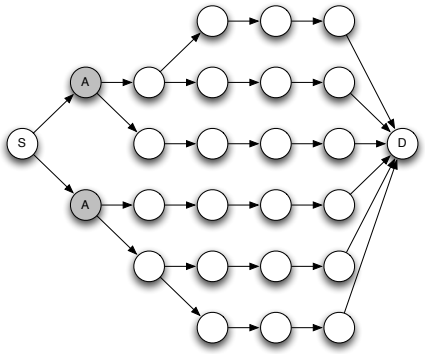
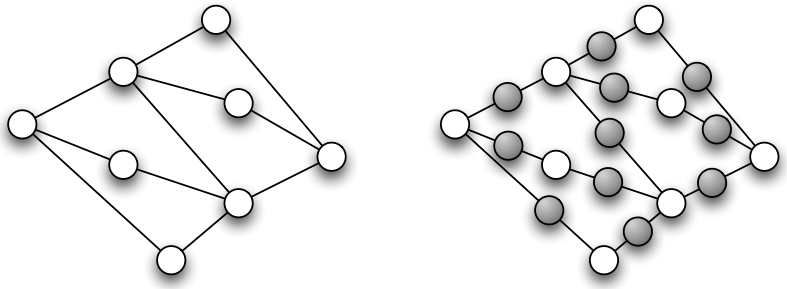


Figure 3.1: A Network where the Threshold Adversary Model is Inappropriate

In the context of multi-party computation, Hirt and Maurer [HM97] proposed another adversary model, the **adversary structure**, which [KGSR02] adapted to network communication. Under this model, the adversary is defined by a set containing all sets of nodes that an attacker can control, and the attacker is assumed to control a single set at any given time. Desmedt *et al.* [DWB06] suggested the **color based adversary** model, where the adversary structure is defined according to the software platform of each node: an adversary that is able to control a certain node running a software platform (one color) is potentially able to control every node with that same platform (same color).

The models described above consider an adversary that controls only devices in a network. In general, these models can be extended to include adversaries also controlling edges by replacing an edge e that connects node A to node B by a new node E and two edges e' and e'' connecting A to E and E to B. Figure 3.2 shows the regular graph representing a network and the transformation of edges in shadowed nodes to adapt it to adversary models based on nodes. One example of an adversary that attacks edges instead of nodes are jammers [DSNW$^+$99], which overload a radio frequency with noise to prevent nodes using that frequency to receive data.

Concerning (ii), three types of behaviors are discussed in the literature [CPA$^+$08]. The first type is a **fail-stop** adversary, that can drop messages that should be forwarded. This adversary does not have access to the content of messages. A **passive** adversary correctly forwards all messages according to the protocol, but is able to read data in the message. The passive adversary is able, for instance, to copy the content of messages to local storage for later analysis. Finally, a **Byzantine** adversary can drop messages in the same way as a fail-stop adversary, can read the content of messages as a passive adversary, but in addition can modify messages, generate more messages, and disrupt the protocol using every other possible strategy.



(a) (b)

Figure 3.2: Extending a Node-based Adversary Model (a) to also Include Networks (b)

Property (iii) determines how the attacker behaves during protocol execution. A **static** adversary selects the set of compromised nodes before the beginning of protocol execution and cannot change the set of compromised nodes while the protocol is running. A **mobile** adversary can change the set of compromised nodes after each step of the protocol, but cannot use information from the current protocol execution to choose the following set of compromised nodes. The most powerful **adaptive** adversary can change the set of compromised nodes on each new step of the protocol, according to information gathered in previous steps of the current execution.

Finally, there are two models for (iv). In the first model the attacker has **bounded** computational power and thus is unable to solve computationally hard problems such as prime factorization or the discrete logarithm, which are the basis for many cryptographic algorithms. In other words, a bounded attacker is unable to reveal an original message from its encrypted version. In the second model the adversary has **unbounded** computational power and protection from such adversaries requires information theoretic protection, also called unconditional protection, which is not based on the difficulty of solving a problem.

HMANETs contain heterogeneous software platforms. Attacks to a software platform have a particularity: once an attack against a specific technology is created, the cost of attacking a single device or all devices using that technology is similar. The cost of attacking another technology, in comparison, is much higher since it requires development of a new attack. Moreover, the risk that a vulnerability is attacked is higher during the vulnerability's attack window, which is the period between discovery of a vulnerability until a correction is available. When a correction is published, the vulnerability can be fixed and exploitation becomes harder. An attack that requires vulnerabilities in multiple platforms is more dangerous during the intersection of the attack window for all vulnerabilities, which is generally smaller than the attack window of each vulnerability individually.

Considering networks, some technologies provide better data protection than others. Wireless connections that use the WPA security protocol provide better confidentiality than an unencrypted wireless connection. Depending on the content of an access, the user may require that the access resists to attacks in specific unsafe network technologies.

Notice that the protocol cannot resist to attacks to the client local network or to the service provider local network since an attacker controlling one of these networks has access to all data the client sends or the provider receives. Unsafe network technologies in intermediary hops can still be avoided with multi-path routing. Our adversary model considers an attacker that is able to control any one software platform available and one network technology from a list of unsafe network technologies.

Another characteristic of HMANETs is that nodes are mostly connected through wireless connections. Wireless networks transmit data through radio signals that can be easily captured by any user authorized or not to access the network. While it is relatively simple for a user to read a message transmitted in the network, it is much harder for unauthorized users to modify messages in the air or to generate additional messages. For this reason, our adversary model considers that attackers are able to control networks with passive behavior representing, for instance, an attacker eavesdropping the wireless network.

Messages transmitted in an HMANET are routed through untrusted user devices. These devices can easily log messages they forward. Modification of message contents or creation of forged messages, though, require modification of the middleware code. As defined in our trust model presented in Chapter 1, we assume that every node runs an authentic version of the middleware. Each node can locally check that the version is authentic, e.g., by verifying a hash code and a signature, and other nodes can verify that neighbor nodes run an authentic version of the middleware, e.g., through an intrusion detection mechanism. In our model, adversaries can also passively control nodes in addition to networks.

Concerning properties (iii) and (iv) discussed above, we assume that attackers can use information obtained during protocol execution to choose a new set of nodes to compromise, and thus are adaptive. We also expect that attackers use computers to analyze data they obtain during protocol execution, so our adversaries have bounded computational power. Formally, consider the network as a graph $G = (V \cup N, E)$ where V represents user devices and N represents the network technology connecting them. Edges are bidirectional and always connect an element from V to an element from N. The adversary can control a structure $\mathcal{A}$ as defined in [KGSR02]. This structure is composed of a set of classes $C = (C_V, C_N)$ where $C_V = \{x | x \in V\}$, $C_N = \{y | y \in N\}$ and thus $C_V \cap C_N = \emptyset$.

3.3 Efficient Multi-path Routing for HMANETs

The protocol for privacy-enhanced service access described in this chapter uses multiple paths between a source and a destination for message transmission. This functionality can only be realized if the underlying routing protocol supports multi-path route discovery and multi-path packet forwarding. In addition, to enable selection of routes that resist to color based adversaries, the routing protocol must provide information about node platform and network technology. In this section we define a multi-path routing protocol that supports message transmission through multiple paths tolerating a color-based adversary that behaves as defined in Section 3.2.

Each network integrated to an HMANET is autonomously organized and runs the routing protocol better adapted to its local requirements. As such, it is unfeasible to define a single routing protocol to be executed by all networks that are part of an HMANET. Rather, a more suitable solution to allow for packet routing among heterogeneous networks is to create an overlay network containing bridges that run a bridge-to-bridge routing protocol while local delivery in each network is performed using the network-specific protocol. Here, thus, we discuss a protocol that explores multi-path routing on the overlay network.

Routing in MANETs is a widely studied problem, and many protocols for efficient routing exist in the literature. There are basically two types of ad hoc routing protocols: **proactive** protocols, such as OLSR [JMC+01], actively exchange control messages to keep an updated view of the network. As nodes moves, the effect of mobility on routes is automatically reflected by the protocol. The alternative approach, **reactive** protocols such as AODV [PR99] and DSR [JMB01], only discover routes when they are needed. Node mobility is not immediately reflected in routes: when an existing route fails, the source node starts a new route discovery to update its routing table. Hybrid protocols such as TORA [PC97], that use reactive routing for nodes communicating less frequently and proactive routing for nodes communicating more often, are also proposed in the literature.

Those protocols, however, are designed to minimize resource usage in mobile devices, and as a result keep a single shortest route between any two nodes in the network. DSR and TORA support multi-path route discovery but only as a mechanism to provide redundant routes when the main route fails. Extensions to AODV such as AOMDV [MD01] or AODV-BR [LG00] also propose to use multi-path routes as backups of the main route. Other approaches suggest the use of multiple routes to improve quality of service: OLSR extensions such as [NM06] and [BMAP03] keep multiple routes to a destination that present optimal quality parameters such as one route with the largest bandwidth and one with the smallest delay. In [KCRI09] we proposed an extension to OLSR that proactively keeps routes with optimal bandwidth, cost and delay, allowing clients to choose routes that optimize the most important of these three properties according to their needs.

Few protocols efficiently provide support for simultaneously using multiple routes in mobile networks. Split Multipath Routing (SMR) [LG01], for instance, is an on-demand routing protocol based on DSR that discovers disjoint routes. Whenever a destination receives multiple route requests, it selects paths that are maximally node-disjoint (i.e., that do not have nodes in common) with the fastest route (the one traversed by the first route request). The node that originated the request, thus, receives a set of destination-selected routes and uses source routing to define the routes that each packet must traverse. MSR [WZSD00] is another multi-path routing protocol based on DSR. Instead of optimizing routing replies as SMR, in MSR all replies are considered and intermediary nodes cache route replies to improve discovery performance. It also gives preference to node-disjoint routes.

As already noted in [DWB06], the issue of finding paths that resist to color based adversaries is not related to finding node-disjoint paths. As a consequence, the routing protocol adopted by our solution cannot restrict route discovery to node-disjoint paths. Moreover, all the protocols described above are designed for MANETs. They assume that

each node has a single network interface and multiple paths among nodes in the network have that interface as origin. In HMANETs, nodes are equipped with various network interfaces and multi-path routing involves activation of different network interfaces. This causes a non-negligible impact on energy consumption that must be considered by the routing protocol. We analyze the relation between routing and energy consumption in Section 3.6.

In addition to the requirements described above, the routing protocol must use source routing. The privacy-enhanced service access protocol proposed in this chapter must be able to send messages to a single destination through different routes. In table-driven protocols, each message contains only its destination address; intermediary routers decide the message's next hop based on their local routing tables. As a consequence, the source node has no control over the path used by a message to get to its destination. Even worse, two messages sent to the same destination will follow the same path (if routes do not change due to node mobility).

We have designed a hybrid ad hoc routing protocol that uses source routes to forward packets and thus ensures that the path defined at the packet source will be respected during packet transmission. The protocol proactively keeps the best route to every node on the network using the efficient flooding mechanism proposed by OLSR [JMC+01] while multiple paths to a destination are discovered on demand by using a technique similar to DSR [JMB01]. Control messages of both protocols contain information about the node's software platform (operating system and version) and the types of network connected to the node (technology, protocol and version).

The reactive component is based on Split Multipath Routing (SMR) [LG01], an extension to DSR that supports discovery of multiple paths between a source and a destination. In SMR, intermediary nodes on a route discovery are not allowed to reply to the source with cached routes to enable the greatest number of ROUTE REQUESTS to reach the destination. As such, the destination learns a bigger number of routes between itself and the source, and can more easily select a set of disjoint routes. Also, to increase the probability of finding disjoint routes, not every duplicate packet is dropped. Instead, only duplicate packets coming from the same neighbor and that have a hop count bigger than the first request are excluded. Experimental results show that this strategy increases the probability of finding routes that are minimally overlapped.

We modify SMR in two ways. First, when forwarding route request messages, an intermediary node adds to the request not only its address but also information about its software platform and network type. Second, we increase the privacy awareness of SMR by modifying its mechanism for route request propagation. A route request in SMR contains not only the source and destination addresses of the route that needs to be discovered, but also the whole path traversed by the packet on its way to the destination. This data unnecessarily reveals to intermediary nodes that a communication will happen between the source and destination of a route request. In our extension, route requests contain only the message's last hop and the number of hops traversed by the message until that moment. To avoid loops, if a node receives route requests with the same identification, it drops those containing a higher hop count. Intermediary nodes keep a list containing

message IDs and the addresses from where they arrived; route replies to each message are sent back to the corresponding address. Only route replies contain the path traversed by the message to inform the source of a route request of the path found.

3.4 Adversary Tolerable Route Selection

The problem of communication in unreliable networks can be considered from two related perspectives [DDWY93]. **Reliable** message transmission is the problem of guaranteeing that a message will reach its destination in spite of an adversary. **Private** message transmission is the complementary problem of ensuring that an adversary will not obtain any information about the message. **Secure** message transmission, then, is defined as the problem of sending a message reliably and privately. In this chapter our main concern is privacy and we consider reliability probabilistically: each path has the same independent failure probability and our protocol uses multiple paths to reduce the probability that a network failure disrupts communication.

The goal of our algorithm for route selection is to choose, among the routes available, a set of routes that can be used to privately transmit a message to the destination tolerating the adversary described in Section 3.2. The early work by Dolev *et al.* [DDWY93] established a number of results on the possibility of secure message transmission considering threshold adversaries. Further research established additional results considering the threshold adversary and different network models (e.g., asynchronous [SKR02], multicast [FW00] or asymmetric [WD02] networks).

Concerning secure message transmission in face of an adversary structure, the work of Kumar *et al.* [KGSR02] defined possibility requirements and a protocol for message transmission in symmetric networks, those where every connection works two ways. These protocols, however, assume Byzantines adversaries and are more complex in terms of computation and communication than the requirements of our adversary model. Those results were extended in [PSC+07] for the case of asymmetric networks. Our approach is similar to the one in [SKR02] as we also define two adversary groups, but in our case both groups have passive behavior while their algorithms assume a group of passive and other of Byzantine behavior. We adapt the algorithm proposed by [KGSR02] to consider only the case of passive adversaries.

Our algorithm considers two adversaries structures. The structure C_V is defined on devices of the network and contains the sets of nodes that an adversary can control when attacking a single available platform. The structure C_N contains the sets of networks that an adversary can control when attacking one network technology among the ones that the user considers unsafe. Our goal is to select a set of routes that resists to the adversary structure $C = (C_V, C_N)$, representing an adversary that controls one set of nodes among the sets of nodes that it is allowed to control and one set of networks among the networks it is allowed to control.

Given an adversary structure $\mathcal{A}$ and nodes S and R, a network is said to be $\mathcal{A}^k(S,R)$-subconnected if the removal of nodes from $\bigcup_{i=1}^{k} A_i, A_i \in \mathcal{A}$ does not disconnect the

network. In other words, S can still communicate with R even when k subsets of $\mathcal{A}$ are removed from the graph. In this setting, if the adversary is passive, secure message transmission is only possible if the sub-network between S and R is $\mathcal{A}^1(S,R)$-subconnected [KGSR02].

We can trivially extend the result above to show that, using the adversary structure defined in Section 3.2 that contains two adversary classes, secure message transmission between S and R is possible if and only if S and R are $\mathcal{A}^{(1,1)}$-subconnected. Suppose that S and R are $\mathcal{A}^{(1,1)}$-subconnected but secure message transmission is not possible. This means that for any $C_i \in \mathcal{A}$, removal of nodes $C_{V_i} \cup C_{N_i}$ does not disconnect S from R. Consider a new adversary structure $\mathcal{A}' = \{F | F = C_V \cup C_N, \forall C = (C_V, C_N) \in \mathcal{A}\}$. Clearly, if S and R are $\mathcal{A}^{(1,1)}$-subconnected, they are also $\mathcal{A}'^1$-subconnected and secure message transmission is possible, which leads to a contradiction.

Conversely, if secure message transmission is possible, there is at least one path that the adversary cannot control. Since the adversary can control any set of nodes in C_V and any set of networks in C_N, there is at least one path connecting S and R that resists to attacks to any set of nodes in C_V and any set of networks in C_N and thus S and R are $\mathcal{A}^{(1,1)}$-subconnected.

The algorithm for route selection defined here produces two outputs: if secure message transmission is possible given the existing topology, the algorithm outputs a set of routes that resists an attacker controlling one color of devices (one software platform) and one color of networks (one network technology). If this is not possible, the algorithm tries to find two different sets of routes: one that resists attacks to devices and other that resists attacks to networks.

Listing 3.1 shows the algorithm, based on the one presented in [SKR02]. In this listing, $\mathcal{A}^X_{acc}$ represents the access structure (the paths free from attackers) when the attacker controls nodes from the set X. We also use $G_{[X]}$ to represent the graph G induced by nodes in X. If S and R are not $\mathcal{A}^{(1,1)}$-subconnected, $Path_{C_V,C_N}(S,R)$ will be empty. The same happens with $Path_{C_V}(S,R)$ if S and R are not $\mathcal{A}^{(1,0)}$-subconnected and $Path_{C_N}(S,R)$ if S and R are not $\mathcal{A}^{(0,1)}$-subconnected.

The privacy guarantees that the protocol provides depend on the network topology. If enough paths exist between source and destination and if they are sufficiently heterogeneous, the user can be sure that communication is private even if an attacker controls all nodes running a software platform and all networks using one technology. If the network topology does not provide one such set of paths, the algorithm still tries to find two sets of paths, one resistant to attacks on nodes and another on attacks on networks. This enables the user – or the middleware on behalf of the user – to evaluate if the trade-off of having less privacy guarantees is desirable.

3.5 Private Message Transmission

After obtaining the multiple paths connecting the source to the destination and selecting a set of routes that resists to attacks from adversaries controlling a structure of nodes and/or

```
// Given G = (V ∪ N, E), A, S and R, the algorithm outputs either the set of paths
// Path_{C_V,C_N}(S,R) or the sets Path_{C_V}(S,R) and Path_{C_N}(S,R)
Let Path_{C_V}(S,R) ← ∅, Path_{C_N}(S,R) ← ∅ and Path_{C_V,C_N}(S,R) ← ∅
Let A_acc^{C_V} ← {A_i = (V \ C_V) | ∀(C_V, C_N) ∈ A} and A_acc^{C_N} ← {A_i = (V \ C_N) | ∀(C_V, C_N) ∈ A}

// Build Path_{C_V,C_N}(S,R)
For i ← 1 to |A_acc^{C_V}|
  For j ← 1 to |A_acc^{C_N}|
  If Path^{G_[A_i ∩ A_j]}(S,R) = ∅
    Path_{C_V,C_N}(S,R) ← ∅ and break;
  Else If Path_{C_V,C_N}(S,R) ∩ Path^{G_[A_i ∩ A_j]}(S,R) = ∅
    Path_{C_V,C_N}(S,R) ← Path_{C_V,C_N}(S,R) ∪ Path^{G_[A_i ∩ A_j]}(S,R)

If Path_{C_V,C_N}(S,R) ≠ ∅
  Return

// Build Path_{C_V}(S,R)
For i ← 1 to |A_acc^{C_V}|
  If Path^{G_[A_i]}(S,R) = ∅
    Path_{C_V}(S,R) ← ∅ and break;
  Else If Path_{C_V}(S,R) ∩ Path^{G_[A_i]}(S,R) = ∅
    Path_{C_V}(S,R) ← Path_{C_V}(S,R) ∪ Path^{G_[A_i]}(S,R)

// Build Path_{C_N}(S,R)
For i ← 1 to |A_acc^{C_N}|
  If Path^{G_[A_i]}(S,R) = ∅
    Path_{C_N}(S,R) ← ∅ and break;
  Else If Path_{C_N}(S,R) ∩ Path^{G_[A_i]}(S,R) = ∅
    Select p(S,R) ∈ G_[A_i]
    Path_{C_N}(S,R) ← Path_{C_N}(S,R) ∪ Path^{G_[A_i]}(S,R)
```

Listing 3.1: Path Selection Algorithm

a structure of networks, the source of the message must use those routes to increase privacy in service access. There are two issues that must be solved in this step: how to divide a message into multiple shares and how to allocate these shares to each route of the selected set of routes.

Many algorithms and protocols exist in the literature that enable a message to be divided into n multiple shares and to be recovered by a participant that has access to t of these shares, with $t \leq n$, but they differ in a number of provided properties. Information dispersal [Rab89], for instance, generates shares with a small overhead while diversity coding [ACLGM93] additionally allows for self-correction. Those algorithms, however, do not provide information security, and a group of shares $u < t$ can reveal some information about the whole message.

Threshold secret sharing [Sha79], on the other hand, generates shares that do not reveal any information about the message even when an adversary has access to u shares, with $u < t$. Traditional threshold secret sharing schemes, however, incur an excessive overhead because in those schemes each share must be at least as big as the message itself [Csi97]. Thus, protocols such as [LLF04] that provide private communication through multiple paths using traditional secret sharing schemes are unsuitable for environments where network is a costly resource, particularly if messages are large.

Traditional secret sharing schemes provide information theoretic secrecy. The attractive solution to avoid their typical overhead in terms of size of the shares is to use a threshold secret sharing scheme that provides computational secrecy such as [Kra94]. This scheme combines the small overhead in message shares from information dispersal algorithms to the secrecy properties of secret sharing. Instead of using the whole message as the secret, the message source generates a symmetric encryption key k and encrypts the message M with this key. The encrypted message M_k is divided into n shares such that t of them are necessary to recover its content using an information dispersal algorithm. The encryption key, generally much smaller than the message, is then used as the secret of a (n, t) secret sharing scheme. The shares distributed through each path contain a share of the encrypted message and a share of the key.

The message size overhead of this protocol is nearly optimal. The overhead in message size of a symmetric encryption algorithm is negligible. The encrypted message is divided by a (n, t) information dispersal algorithm, which provides optimal share size. In other words, if the message M_k can be recovered from t shares, each share has a size of $\frac{|M_k|}{t} \approx \frac{|M|}{t}$. Considering AES as the symmetric encryption algorithm, a key size of 128 bits provides an adequate level of protection, so we can consider that the key length $|k| = 128$ bits, and each share of the key has also 128 bits. The total size of each share generated by this algorithm, thus, is $\frac{|M_k|}{t} + |k|$ which is approximately $\frac{|M|}{t} + 128$, or 128 bits longer than the optimal size. Notice however that the size of each share is still dependent on the threshold, or in other words, the number of shares required to recover the message.

Our protocol is based on the one defined by Dolev *et al.* [DDWY93] but we introduce two modifications. Since in our adversary model an attacker controlling nodes and networks cannot modify the content of messages, our protocol does not require error-correction mechanisms for the detection of faulty shares. This reduces the amount of

additional information added to each share and thus optimizes network usage.

In addition, the original protocol considers that an adversary can drop messages as part of an attack. We believe that adversaries have no interest in dropping messages and revealing themselves to the rest of the network, so we consider that transmission errors occur only as a result of network failures and we treat transmission errors probabilistically. Each route has an independent probability $(1-\phi)$ of failure, that can be determined as the average message delivery failure rate observed by the middleware. The algorithm considers a user-required reliability ρ and defines the threshold of the information dispersal algorithm as the largest number of routes that still achieve the user-required reliability ρ.

To understand why we use this process for defining the threshold of the secret sharing scheme, imagine that there are n routes available between the source and the destination. The highest reliability can be obtained by using a $(n, 1)$ secret sharing scheme, since in this case only one share from the n shares transmitted is required to recover the whole message. The failure probability, thus, is $(1-\phi)^n$, or in other words the destination is not able to recover the message only if all routes fail.

The side-effect of choosing 1 as the threshold for a secret sharing scheme with computational secrecy is that every message share in this case must be as large as the message itself, and the total overhead will be $n \times |M|$. On the other hand, a large threshold value reduces the size of each share and the total overhead, but also reduces reliability since it requires a larger number of routes to work to not compromise message delivery. As a result, our strategy is to maximize the threshold while still maintaining the user-required reliability to reduce total overhead as much as possible without jeopardizing reliability.

Before creating message shares, the middleware must define the protection level required according to the paths available on the environment. This decision can be based on a user profile or asking for the user's intervention. If the sender wants protection against eavesdroppers on both networks and devices and the network topology supports it, the algorithm uses paths in the set $Path_{C_V,C_N}(S,R)$ created by the protocol in Section 3.4. If it is only interested in protection against passive attackers on nodes (respectively edges), it uses the paths in $Path_{C_V}(S,R)$ (respectively $Path_{C_N}(S,R)$).

Let $Path(S,R)$ be the set of paths the sender selects for transmission and $P(S,R)$ be the set of all paths in $Path(S,R)$, or $P(S,R) = \{p | p \in P_i, P_i \in Path(S,R)\}$. Consider $n = |Path(S,R)|$, or in other words the number of sets of paths selected by the route selection algorithm, and $n' = |P(S,R)|$, or in other words the number of different routes in the sets of routes selected by the route selection algorithm. The source node must define two threshold values: t represents the number of shares required for the destination to recover the key and t' represents the number of shares required to recover the encrypted message.

By construction we know that there is at least one set of paths in $Path(S,R)$ not controlled by the adversary, and the adversary can control at most $n-1$ sets in $Path(S,R)$. To ensure that the adversary is unable to recover the key, we define $t = n$. In order to reduce the size of shares without compromising reliability, we define t' as the largest value providing probability of successful transmission at least as high as the user required success probability.

The sender defines a random symmetric encryption key k and encrypts the message M using this key with the AES encryption algorithm to obtain $M_k = AES(M, k)$. Using Rabin's information dispersal algorithm [Rab89] and values n' for the total of shares and t' for the number of required shares to obtain the message, the sender splits the message into n' shares $E = (E_1, ..., E_{n'})$. It also defines a random polynomial $f(x) = k + a_1x + ... + a_nx^{(n-1)}$, and random values $(\alpha_1, ..., \alpha_n)$ to obtain key shares $F = (F_1, ..., F_n)$, where $F_i = (\alpha_i, f(i))$.

For each path p_j with $1 \leq j \leq n'$ in each set $P_i \in Path(S, R)$ with $1 \leq i \leq n$, the sender sends a share $S_{ij} = (F_i, E_j)$. In other words, all routes in the same set of routes receive the same share of the key. Since an adversary cannot control all sets of routes at the same time, there is at least one share that the attacker is not able to control. Every route, regardless of the sets of which it is member, receives a different share of the encrypted message. Since the message is encrypted, even if an attacker is able to obtain more shares than the threshold it is not able to understand its contents.

The routing protocol presented in Section 3.3 does not guarantee that a sender and a receiver share the same view of the network. Indeed, whenever a sender requires multiple paths for a communication, it starts a route discovery and builds a network map from the responses. The receiver replies to a number of route requests but is not aware of all intermediary hops on paths to the receiver. As a result, in order to recover the original message, the receiver must be informed on the total number of shares that the sender creates. Each share thus, contain also the number of shares required to recompose the key $t = n$ and to recompose the encrypted message t'.

There are protocols in the literature that enable a sender and a receiver to identify malfunctioning paths after a few communication rounds [DDWY93, KGSR02]. We believe that the extra communication cost is not worth the benefit of detecting bad paths, particularly in mobile networks where connections are unstable. For that reason, in our protocol the receiver sends only two types of reply: either OK if it was able to reconstruct the secret from the shares received, or ERROR in the opposite case. The reply is sent through all paths from which the receiver obtained shares. If the sender receives an ERROR, it generates a new encryption key and a new polynomial and restarts the protocol. Listing 3.2 shows the complete algorithm for private and probabilistically reliable service access.

It is important to notice that the receiver may obtain the same key share from different paths and more than one key share from the same path. This is because the sets in $Path(S, R)$ may have more than one element and they may overlap, so the sender can send the same share through different paths and different shares through the same path. Even though this increases the communication overhead of the protocol, it increases reliability of key transmission. A further extension to reduce the overhead of this protocol would be to consider the reliability of each path and only send the same key share redundantly through multiple paths of a set when they provide small reliability.

If message transmission fails, the sender runs the protocol again. Since each run of the protocol is independent and the polynomials are selected at random, the shares generated on the first protocol execution are totally unrelated from shares generated in

// Given a message M, a key k a set of paths $Path(S,R)$, user
// defined success probability ρ and average path reliability ϕ,
// privately sends M with success probability $\geq \rho$ if possible

// Sender:
Let $n \leftarrow |Path(S,R)|$
Let $n' \leftarrow |\bigcup_{i=1}^{n} P_i|, P_i \in Path(S,R)$
Let $m \leftarrow 1$
While $m \leq n'$ do
$\sigma_m \leftarrow \sum_{i=m}^{n'} \binom{n'}{i} \phi^i (1-\phi)^{n'-i}$
If $\sigma_m \leq \rho$ let $t' \leftarrow n' - m + 1$
Else let $m \leftarrow m + 1$
If $t' = n'$ return maximum available reliability
Encrypt M with k and let $M_k = AES(M,k)$
Use Rabin's IDA algorithm on M_k to obtain $E = (E_1, ..., E_{n'})$ in such a
way that M_k can be recovered from t' or more shares
Choose a random polynomial $f(x) = k + a_1 x + ... + a_{n-1} x^{n-1}$
Choose a set of random numbers $\alpha = (\alpha_1, ..., \alpha_n)$
Let $F \leftarrow ((\alpha_1, f(\alpha_1)), \ldots, (\alpha_n, f(\alpha_n)))$
For each $P_i \in Path(S,R)$
For each $p_j \in P_i$
Send (F_i, n, E_j, t') over p_j

// Receiver:
Let $T \leftarrow (T_1, ..., T_m)$ be the shares received from paths $(q_1, ..., q_m)$, $T_i = (U_i, n, V_i, t')$
If $m < n$ or $m < t'$
Send ERROR on every q_i
Else
Using Rabin's IDA, recover M_k from $V = (V_1, ..., V_m)$
Interpolate $f(x)$ from $U = (U_1, .., U_m)$
$k \leftarrow f(0)$
Decrypt M_k with key k to obtain $M = AES^{-1}(M_k, k)$
Send OK on every q_i

Listing 3.2: Reliable Message Transmission Protocol

further executions. As a result, since controlling a number of shares smaller than the threshold does not give the adversary any information about the message, an adaptive adversary cannot obtain more information by changing the set of controlled nodes between executions than it would obtain controlling a static set of nodes.

3.6 Experimental Results

As we mentioned in Section 3.3, multi-network routing causes a non-negligible effect in energy consumption for two reasons. First, a routing protocol running on multiple network interfaces requires activation of additional interfaces that are possibly not being used by the device owner. While in traditional MANETs the same network interface used for service access is also used for sending routing control messages, in HMANETs routing through multiple networks causes an additional energy consumption overhead since activating extra network interfaces and keeping them connected to a network consumes the device's battery on a task not directly related to the user goal.

The second reason why multi-network routing causes a non-negligible impact on energy consumption is that simultaneous usage of multiple radio networks can cause interference among networks. Interference happens if interfaces use the same technology (and thus the same radio frequency) or not. For instance, Bluetooth can interfere with WiFi networks [GDS01]. Because of radio interference, the number of unsuccessful packet transmission increases and network throughput is reduced. As a consequence, data transmission takes longer and consumes even more battery.

The goal of our evaluation is to understand the effects of multi-radio networking on energy consumption to assess the feasibility of multiple network routing. The results of these experiments also helped us on the design of the routing protocol presented in Section 3.3. The protocol aims at reducing the energy impact of providing multi-radio routing in HMANETs, and these results indicates the benefits of a hybrid approach.

We use two handhelds, each featuring a WiFi interface and a Bluetooth interface, to measure the power consumption of multiple interface networking. One device, that acts as the source of route requests and sends messages, is an HP iPAQ 114 Classic with a Marvell PXA310 processor running on 624MHz and 256MB of ROM plus 64 MB of RAM. The other device, that only receives packets, is an HP iPAQ hw6915 with a PXA270 processor running on 416MHz and 128MB of ROM plus 64MB of RAM.

For these tests, we suppose that the user of a pervasive computing system normally enables a single interface for service access and we want to measure the energy overhead of enabling an additional interface for routing traffic from other peers. This is the worst case for multi-interface routing in pervasive networks because it assumes that the user only needs to communicate through a single network interface, and all the energy consumed by the secondary interface is an overhead. In reality, users will often have more than one network interface activated to perform their own tasks, in which case the overhead would only concern additional routing control messages required to enable routing through those interfaces.

As a baseline, we measure the energy consumption of having a single interface enabled and transmitting data. After that we measure the energy impact of activating an extra interface in 3 different situations: when the secondary interface is connected to the network but silent (not transmitting packets), when it is connected and transmitting packets with low frequency (one every 60 seconds) and when it is connected and transmitting packets with high frequency (one every 2 seconds). Both situations where the secondary interface is transmitting data aim at simulating routing protocols running on an additional interface. We send 50 bytes packets which is the average size of a HELLO message in OTRS or a RREQ message in DSR. The interval of 2 seconds is the interval recommended by the OTRS RFC between HELLO messages but also represents a reactive protocol where a new route request is created every 2 seconds, while the 60 seconds interval simulates a reactive protocol where a new route must be discovered every minute.

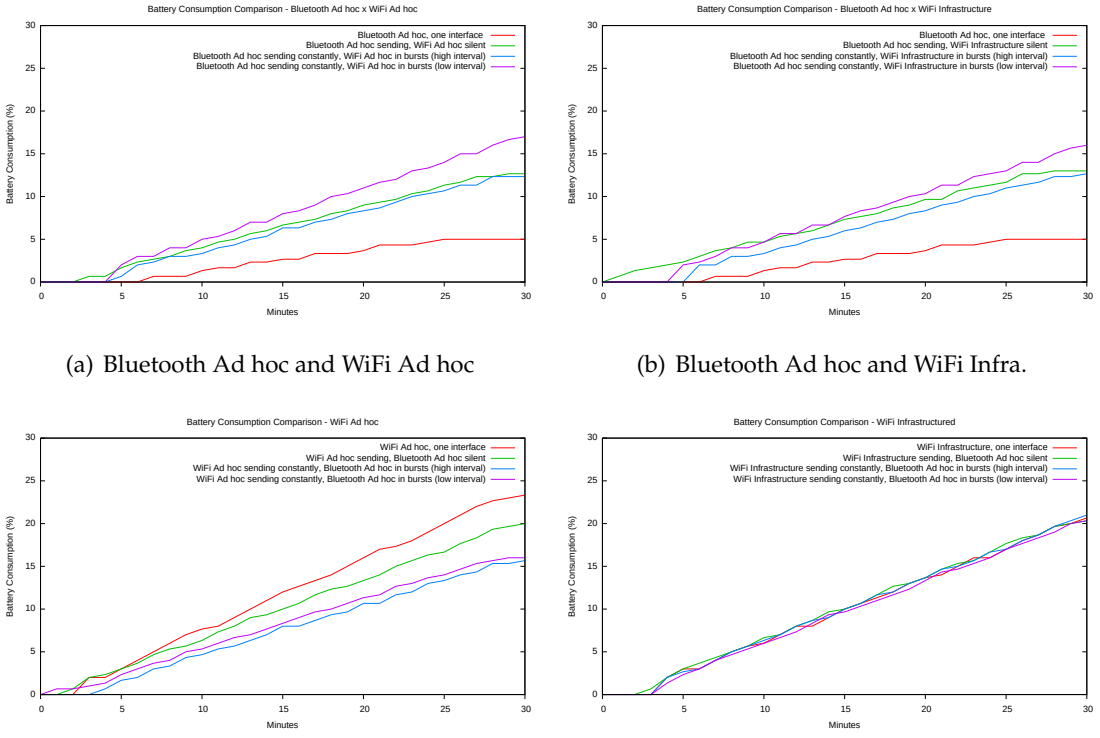
Primary Interface	Secondary Interface
Bluetooth Ad Hoc	WiFi Infrastructure
Bluetooth Ad Hoc	WiFi Ad hoc
WiFi Infrastructure	Bluetooth Ad Hoc
WiFi Ad Hoc	Bluetooth Ad Hoc

Table 3.1: Configurations Used for Energy Impact Evaluation of Multi-Interface Networking

Wireless networks generate low-level control packets to keep the network organization. The quantity of packets generated by an infrastructure and an ad hoc network is different, and so is their overhead in terms of energy consumption [CIC08]. We consider both types of network for each technology we evaluate. Table 3.1 summarizes the test cases considered in our experiments. For each of these cases we simulate the three aforementioned situations: when the interface is silent, when it is transmitting packets with low frequency and when it is transmitting packets with high frequency.

The client device constantly transmits to the server device a file of 32KB during 30 minutes on the main interface. Each test is performed three times with the screen luminosity set to a minimum to restrict energy consumption only to networking operations. After each test, the device's battery is completely recharged before running the following test. The WiFi infrastructure network is dedicated to this experiment to reduce the influence caused by other computers accessing the same network and overloading the radio frequency.

Figure 3.3 shows the impact on energy consumption of using an additional network interface for each configuration evaluated. We can see from graphs (a) and (b) that, if the user's main interface uses Bluetooth technology, activation of an additional WiFi interface more than doubles the energy consumption. This is expected since Bluetooth networks are short-range and require less energy to transmit packets than WiFi. More importantly,



(a) Bluetooth Ad hoc and WiFi Ad hoc

(b) Bluetooth Ad hoc and WiFi Infra.

(c) WiFi Ad hoc and Bluetooth Ad hoc

(d) WiFi Infra. and Bluetooth Ad hoc

Figure 3.3: Energy Consumption on Each Interface Configuration

if we consider that the WiFi interface is already on, the impact on energy of sending a few packets or not sending any packets at all is similar. Whenever a WiFi interface is already connected, the impact of sporadical route requests is negligible.

Graphs (c) and (d) show the energy consumption when the primary interface uses WiFi technology. In that case, the energy costs of having a Bluetooth interface also connected to a network and transmitting packets is very low. If the WiFi interface is connected to an infrastructure wireless network, the impact of activating a Bluetooth interface and sending route discovery packets is negligible, even if those packets are sent frequently. However, if the WiFi interface is connected to an ad hoc network, the energy consumption with a Bluetooth interface activated and transmitting is smaller than with the WiFi interface alone. This result may be unexpected at first, but it is explained by the interference caused by Bluetooth over WiFi, as it becomes clear with the following graphs.

Figure 3.4 shows the effect on the primary interface throughput when the secondary interface is connected. We can see in this graph that even though in all configurations the additional interface reduced the throughput of the primary interface, this impact is stronger when the primary interface uses the WiFi technology configured in the ad hoc mode. We believe this happens because infrastructure WiFi networks have algorithms that efficiently handle packet collisions while packet collision in ad hoc networks is harder

to handle and causes more energy consumption.

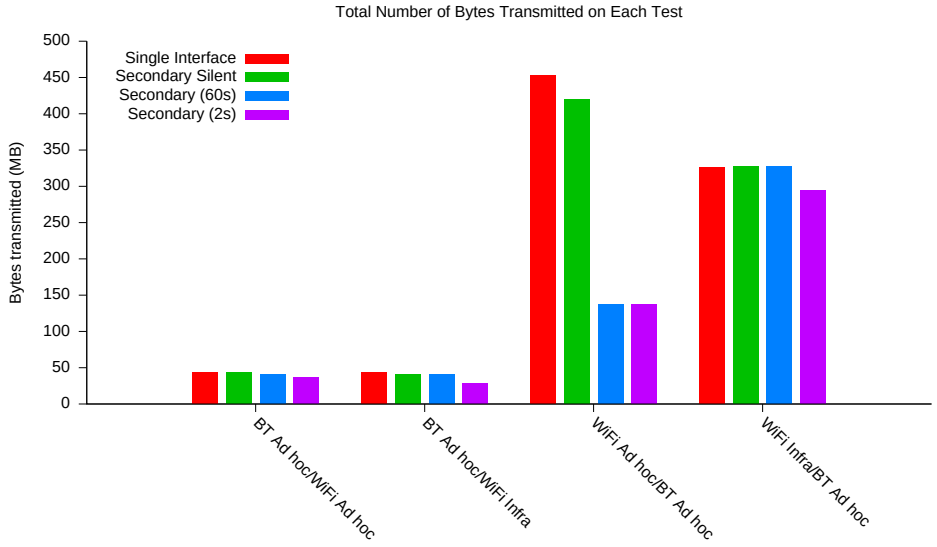


Figure 3.4: Main Interface Throughput

Figure 3.5 shows the relation between bytes transmitted and energy consumption in various configurations. In this graph it becomes clear why it is not advantageous in terms of energy consumption to activate a Bluetooth network interface when the primary network is WiFi in the ad hoc mode: even though the energy consumption is smaller, the main interface's throughput is smaller and finally the relation between bytes transmitted and energy consumed is not advantageous.

We can conclude from this numbers that, whenever the user is using a single WiFi interface in the infrastructure mode for accessing services in a pervasive environment, another Bluetooth interface can be activated without causing a noticeable increase in energy consumption. If the primary interface is WiFi configured in the ad hoc mode, the additional Bluetooth interface can be activated only if the main interface is not being used for data intensive applications (since simultaneously using a Bluetooth interface considerably reduces the primary's interface throughput). Finally, if the main interface is Bluetooth, activating a WiFi interface has a noticeable impact on energy. However, if the user already has both interfaces active, the energy impact of transmitting routing control messages on the additional interface is negligible.

It is important to notice that while existing mobile devices already feature multiple network interfaces with different technologies, such as Bluetooth, WiFi and 3G/UMTS, the software layer support for multi-radio networking is still under development and in some cases it does not allow for simultaneous usage of network connections, e.g., a 3G/UMTS and a WiFi interface. This particularity of current mobile devices complicates practical evaluation of algorithms that depend on simultaneous usage of multiple network interfaces. We believe that the evolution of mobile operating systems and open source initiatives such

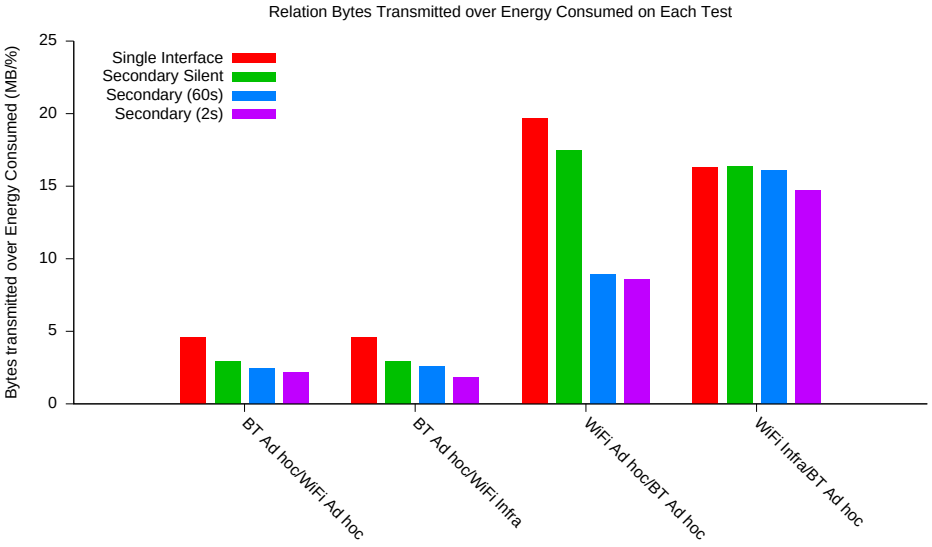


Figure 3.5: Relation Between Throughput and Battery Usage

as Android are already contributing to overcome this limitation, and in the near future we will be able to conduct real world evaluations of multi-radio protocols.

At the same time, current network simulators have limited support for multi-radio networking. Established simulators such as ns-2[6] do not provide standard support for multi-radio networks. Proposed extensions to enable multi-network simulation in ns-2 are not standardized, complicating comparison of results obtained through different extensions. This is undesirable since one of the main goals of a discrete simulation is to facilitate comparison of results obtained with different protocols. Moreover, existing extensions to ns-2 allow only creation of nodes with multiple interfaces that use the same network technology, which is not the case in HMANETs.

Recent efforts to build a new and more easily extensible simulator such as ns-3[7] still do not support Bluetooth or 3G/UMTS networks. Alternatives to ns-2 such as GloMoSim[8] or JiST/SWANS[9] also do not support wireless interfaces that use standards different than those based on the IEEE 802.11 standard. However, multi-radio support for network simulators is also an active area of research and we are confident that simulators supporting multiple radio interfaces will become available in the near future for a more detailed evaluation of the protocol described in this chapter.

[6] isi.edu/nsnam/ns
[7] www.nsnam.org
[8] pcl.cs.ucla.edu/projects/glomosim
[9] jist.ece.cornell.edu

Listen,

Do you want to know a secret?

Do you promise not to tell?

The Beatles, "Do You Want to Know a Secret"

4

Privacy-enhanced Service Discovery

Pervasive computing environments are very dynamic. Mobile users join and leave the system frequently and create networks ad hoc. Applications in pervasive computing must cope with this unpredictability and adapt to the constant changes in the environment. Service discovery provides the main mechanism for client adaptation in a Service Oriented Architecture. Whenever a service fails or disappears, clients use the service discovery mechanism to find another compatible service to replace it. When a client arrives at an unknown environment, it uses service discovery to identify available services. Service discovery contributes to reduce the coupling between clients and service providers and is essential to achieve the flexibility proposed by SOAs.

Discovery information, however, is sensitive and must be protected. Service discovery is generally used before an access, so discovery data may reveal information about the services a user accesses. Discovery requests may also contain additional quality information to find services most suited to the client needs, and may reveal private data as well. As a consequence, an external observer that analyzes service discovery traffic can infer a number of private information about the protocol participants.

In this chapter we present our solution for a privacy-enhanced service discovery protocol, called EVEY. We start by briefly introducing the existing protocols for service discovery in pervasive computing in Section 4.1, where we detail syntactic (Section 4.1.1)

and semantic (Section 4.1.2) service discovery protocols and propose a general model for service discovery protocols (Section 4.1.3) that we later use to detail EVEY. After that we discuss the service discovery impact on privacy in Section 4.2, and define a privacy attack model for service discovery in Section 4.2.1. To motivate our work, we suggest two naive approaches for privacy protection in service discovery protocols and discuss why they are unsuitable for service-oriented pervasive computing in Section 4.2.2.

After introducing a model for service discovery protocols and discussing privacy issues of service discovery, we describe two protocols for privacy-enhanced service discovery in Section 4.3 that provide different types of privacy guarantees but that also have different requirements in terms of infrastructure: the first protocol, described in Section 4.3.1, does not require that clients and service providers share any secret before running the protocol but provides weaker privacy guarantees, while the second protocol, described in Section 4.3.2, provides stronger privacy but requires that clients and service providers share a secret, and thus demands additional infrastructure to enable key exchange. Both protocols are evaluated in terms of privacy protection and performance in Section 4.3.3. An initial version of EVEY was presented in [CBUI07] and demonstrated in [BRC+07].

EVEY provides privacy-enhanced for service discoveries performed inside a single network domain. However, service discovery in pervasive computing usually involves multiple domains, with hierarchies of service directories, and service discovery requests are distributed to remote domains. In Section 4.4 we discuss the privacy risks introduced by distributed service discovery. We propose an extension to EVEY for reducing these risks in Section 4.4.1 and present an assessment of the mechanisms introduced in Section 4.4.2.

4.1 Service Discovery Protocols

One of the greatest benefits brought by SOAs is the potential of having loosely-coupled clients and service providers. This means that developers can create distributed applications where clients are not bound to a specific service implementation deployed at a given location. Rather, clients can adapt to changes on the environment and bind to available services at run-time. A consequence of this reduced coupling is that changes in services have a smaller impact on clients and vice-versa. For instance, a service under heavy load can be easily replicated without the burden of updating client references. Loosely-coupled applications are more flexible and scale better than applications that are strongly tied to clients. Service discovery provides to clients the means to find relevant available services and to bind to their interfaces, enabling service providers and clients to find each other. Service discovery protocols are a critical component of an SOA since they are used to bootstrap interactions between clients and services.

There are many ways that service providers can use to communicate the existence of a service and its interface to clients, and not surprisingly a number of protocols exist that use different approaches for service discovery. Zhu *et al.* [ZMN05] propose a comprehensive classification of service discovery protocol features in pervasive computing. Concerning the protocol required infrastructure, they suggest that directory-based protocols – where

a repository accumulates service announcements and answers to service requests – are more efficient in environments with a larger number of devices or services than nondirectory-based protocols – where providers and clients broadcast or announce on a multicast channel service announcements and requests.

In this thesis we focus on directory-based service discovery protocols. In such protocols, participants can play three different roles: **client**, **service provider** and **service directory**. The protocol generally comprises four steps, namely **publication**, **location**, **matching** and **selection**. Three types of service description are handled by the entities: **announcements**, **requests** and **results**. Figure 4.1 shows how those components interact. Participants are depicted as rounded-corner rectangles, and a rectangle inside a participant represents a process that is performed locally by that participant. The messages they exchange are represented by arrows associated to the respective types of service descriptions contained in each message.

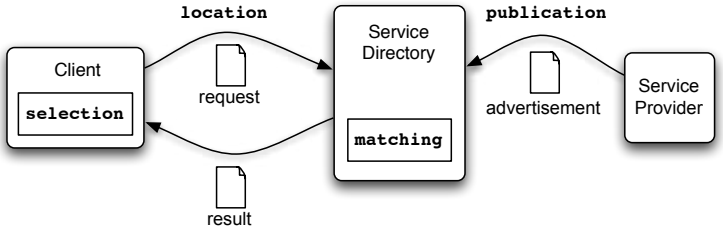


Figure 4.1: A Directory-based Service Discovery Protocol

The efficacy of a service discovery protocol depends on its matching, or matchmaking [LH04] algorithm, which is responsible for comparing requests with announcements to find compatible services. The quality of a matching algorithm can be measured by two metrics: the number of irrelevant descriptions included in a response to a discovery request, and the number of relevant descriptions not included by the algorithm in a response to a discovery request. A good matching algorithm minimizes both metrics: its results contain only relevant services and all relevant services are returned.

The ability that a matching algorithm has to identify relevant services is directly related to the expressiveness of the service description language it uses. Client requirements and service properties may involve subjective criteria and requires definition of multiple characteristics of the service. An expressive language enables clients and providers to characterize services with a higher level of detail and provides them the flexibility to describe their needs and their offers so as to increase the probability of a match.

Existing service description languages can be divided into two large groups: languages that accept only syntactic descriptions of service elements, and languages that also provide means to semantically describe those components. The following sections discuss the two main classes of service discovery protocols: Section 4.1.1 describes the features of syntactic protocols, that can be considered as the first generation of service discovery protocols, while Section 4.1.2 examines semantic protocols, that apply the ideas of the Semantic

Web [BLHL01] to service discovery to convey more meaning to service descriptions. After presenting these two types of protocols, we present a general model for describing service discovery protocols that is later used to characterize EVEY.

4.1.1 Syntactic Service Discovery

Early protocols proposed for service discovery consider service descriptions as a set of strings, eventually conforming to a document structure, which specify the service features. Those protocols are considered syntactic because their matching algorithms syntactically compare service descriptions to determine if there is a match. If the strings contained in the service request and in the service announcement are slightly different, there is no match. If the strings are identical but actually represent different concrete objects, the matching algorithm gives a positive result.

There is a number of syntactic service discovery protocols proposed in the literature, so we will focus our discussion only on the most relevant ones. One example is the Service Discovery Service (SDS) [CZH+99]. In SDS, service announcements (called "service descriptions") and service requests (or "service queries") are described using XML. The file format is not completely defined by the protocol: some tags are common to all services (such as service location) but service-specific tags can be created to define additional service capabilities.

Sun proposed Jini as a discovery protocol for Java applications [Wal99]. Applications developed using the Java language can export a reference to a Jini lookup service containing $< property, value >$ strings and an RMI proxy reference to the service. Clients also developed in Java can programatically access the lookup service to locate services using $< property, value >$ pairs as well. Additionally, Jini provides code mobility features that enable clients to dynamically download the code necessary to access a service.

Another popular syntactic service discovery protocol is Universal Plug and Play, or UPnP [JW03], originally proposed by Microsoft and later standardized by the Universal Plug and Play Forum. Services using UPnP are described through an XML file that specifies their characteristics. UPnP, however, does not rely on a service directory, and services can broadcast their announcements to the network as well as clients can broadcast their requests. To reduce broadcast overhead, service announcements and requests contain only a few details and a pointer to a more complete XML description. Clients can later parse the XML description referenced by a service announcement and syntactically compare tags and values in the announcement with the service properties they are looking for.

In the context of Web services, two protocols for service discovery were proposed. One is the Universal Description, Discovery and Integration protocol, or UDDI for short [OAS05d]. It defines a flexible data format for the exchange of service descriptions and a network protocol that enables announcements and queries for services. Service providers describe their services using the standard UDDI XML format for service descriptions and publish them in UDDI registries. Clients can find services either manually browsing the registry or using the UDDI SOAP API to query UDDI registries. Queries can be performed by service type, company, or based on the service's access level details. The registry syntactically

compares query strings to service announcement strings to find matches.

The second protocol proposed for Web service discovery is WS-Discovery [B+05a]. While UDDI focuses on more complex networks and a centralized repository, WS-Discovery is a lightweight protocol for ad hoc networks. It does not require a service directory, and service providers announce services to a multicast channel while clients send service probes to the same channel. Announcements and probes contain very basic service information, and a match is considered whenever the strings are syntactically equivalent. Nodes can also volunteer to be a Discovery Proxy (DP), in which case announcements and probes are transmitted to the DP through unicast.

Syntactic service discovery protocols have a number of advantages. They are usually very efficient in terms of processing since they are based on string comparisons, which can be efficiently implemented. They are also simple to deploy and enable clients and service providers to describe services with an acceptable level of details. Nevertheless, syntactic service discovery protocols are inappropriate for open environments such as pervasive computing. Even though some efforts exist for service interface standardization, it is not reasonable to expect that a common interface will be standardized for every possible pervasive service. UPnP, for instance, proposes that services are specified through standard Device Templates, defined by the UPnP Forum. These templates describe standard functionalities for classes of devices, and can be extended by each device manufacturer. It is very unlikely that all the extensions will also follow a standard, and this approach does not scale when the interface of user-provided software services must also be standardized.

As a consequence, rarely the client-specified service interface syntactically matches the one that the provider announces. More likely, service providers will specify service interfaces using their particular vocabulary while clients will describe services using their own terms since they do not know each other in advance and cannot agree on a common vocabulary for service description. This characteristic of service discovery in pervasive computing considerably reduces the effectiveness of syntactic protocols, as they are unable to find similarities between descriptions if these protocols are slightly different syntactically.

Still, syntactic service discovery must be supported in pervasive computing. Resource-constrained devices may implement only syntactic discovery protocols for efficiency reasons, or legacy resources on a pervasive environment may run a syntactic service discovery protocol because it was the single alternative at the design time. Pervasive computing middleware, thus, must support syntactic service discovery in spite of their limitations when clients and service providers do not agree on interface descriptions.

4.1.2 Semantic Service Discovery

To overcome the weaknesses of syntactic service discovery protocols, researchers proposed the creation of service description languages with greater expressiveness, which resulted in the creation of semantic service discovery protocols. Such protocols enable syntactic service descriptions to be extended with semantic annotations. The semantic matching algorithm later takes into account the semantic annotations to define a degree of match between a service request and a service announcement.

In a semantic service description, elements are augmented with semantic concepts that are part of an ontology. An ontology is a formal definition of the relation among concepts of a particular domain [BLHL01]. The matching algorithm uses the ontology structure and the concepts in semantic service descriptions to verify if concepts are related, how they are related and how similar are their meaning. Those relations enable the matcher to identify service descriptions that are not syntactically identical but are annotated with concepts that express similar ideas.

There are two complementary approaches to semantic matching of service descriptions [CF03]. The first is called **signature matching** and its goal is to find if the signature of an advertised capability (for instance its inputs and outputs) is related to the signature of a requested capability. There are three types of relations that a matchmaking can find [PKPS02]: **exact** when a request matches exactly an announcement, **plug-in** if the announcement is more generic than the request and **subsumes** if the request is more generic than the announcement. Approaches such as [PKPS02, TBG01] are examples of signature matching algorithms.

The second approach is **specification matching**, and instead of relying on capability signatures to define if announcements and requests are related, specification matching determines if the behaviors of advertised and requested services are the same. Specification matching algorithms require services to specify the conditions that hold after service execution (the post-conditions) given that a set of conditions was valid before service execution (the pre-conditions). The algorithm creates predicates based on pre- and post-conditions, and compute the relation between service announcements and service requests according to service predicates. Techniques to compute this type of matching relation includes theorem-proving [ZW97] and θ-subsumption [SYKW99].

Two service descriptions are said to semantically match if both their signature and their specification match [ZW97]. However, specification matching is resource intensive and has infrastructure requirements incompatible with pervasive environments [Ben07]. Signature matching, on the other hand, also consumes resources but recent work proposed mechanisms to optimize matching performance [CF03, BKGI06], making signature matching an adequate approach for semantic service discovery in pervasive computing.

The EVEY protocol for privacy-enhanced service discovery presented in this chapter supports both semantic and syntactic service descriptions. In EVEY, we assume that there is no universal agreement over the ontology used to describe a certain domain of interest. Service developers can use different ontologies because even though two ontologies may describe the same domain, each one may emphasize a different aspect of the domain under consideration, or because domain specialists may disagree on the number of concepts and their relation, which leads to development of distinct ontologies describing the same domain. Finally, an ontology definition may be too large for some applications, so ontology developers may divide one large ontology into smaller parts describing specific aspects of a domain to be used in applications that do not have to manipulate information concerning all domain-related areas.

4.1.3 A Model for Service Discovery Protocols

In order to properly compare the characteristics of existing service discovery protocols with our solutions, and to identify how EVEY modifies existing protocols, we need a model to describe syntactic and semantic service discovery protocols in general. A service discovery protocol can be defined as:

- A language $\mathcal{S}$ for describing a service announcement composed of a set of symbols $\mathcal{A}_\mathcal{S}$ and accepted sequences $\mathcal{F}_\mathcal{S}$. $\mathcal{S} = \{\mathcal{A}_\mathcal{S}, \mathcal{F}_\mathcal{S}\}$;
- A language $\mathcal{R}$ for describing a service request composed of a set of symbols $\mathcal{A}_\mathcal{R}$ and accepted sequences $\mathcal{F}_\mathcal{R}$. $\mathcal{R} = \{\mathcal{A}_\mathcal{R}, \mathcal{F}_\mathcal{R}\}$;
- A matching function $\mathcal{M}$ that computes the distance between a service announcement and a request, assigning to a pair of elements from $\mathcal{S}$ and $\mathcal{R}$ another element of a set D. $\mathcal{M} : \mathcal{S} \times \mathcal{R} \rightarrow D$.

In traditional syntactic matching, languages $\mathcal{S}$ and $\mathcal{R}$ are the same, and the alphabets $\mathcal{A}_\mathcal{S}$ and $\mathcal{A}_\mathcal{R}$ contain only syntactic symbols without meaning. As such, a matching algorithm comparing sequences $w_S \in \mathcal{F}_\mathcal{S}$ and $w_R \in \mathcal{F}_\mathcal{R}$ can only have two possible outcomes: either $w_S = w_R$ or $w_S \neq w_R$. We consider $w_S = w_R$ if all their symbols are identical in the same order.

Services are described as a set of required inputs I_S, a set of provided outputs O_S and a set of provided properties P_S. Service descriptions may also contain access related information A_S that is not used by the matching algorithm, $S = \{I_S, O_S, P_S, A_S\}$. Each input $i_S \in I_S$ have a name n_{i_S}; outputs $o_S \in O_S$ and properties $p_S \in P_S$ are similarly defined as $o_S = \{n_{o_S}\}$ and $p_S = \{n_{p_S}\}$. A property is any feature that helps to describe a service but is not related to its input or output parameters, such as the service name or the service category. We define a syntactic matching function as $\mathcal{M}_{syn} : \mathcal{S} \times \mathcal{R} \rightarrow D$ with $D = \{0, \infty\}$ and we say that for a description $S \in \mathcal{S}$ and a description $R \in \mathcal{R}$, if $\mathcal{M}_{syn}(S, R) = 0$, service S syntactically matches request R, while if $\mathcal{M}_{syn}(S, R) = \infty$ the service and the request do not match.

Semantic matching introduces ontologies in the languages for service announcement and service request. In a semantic service discovery protocol, $\mathcal{S} = \{\mathcal{A}_\mathcal{S} \cup C(\mathcal{O}), \mathcal{F}_\mathcal{S}\}$ and $\mathcal{R} = \{\mathcal{A}_\mathcal{R} \cup C(\mathcal{O}), \mathcal{F}_\mathcal{R}\}$ where $\mathcal{O}$ is the set of all existing ontologies and $C(\mathcal{O})$ is the set of all concepts defined in $\mathcal{O}$. These concepts can be combined to other symbols of the language to attach meaning to sequences in $\mathcal{F}_\mathcal{S}$ and $\mathcal{F}_\mathcal{R}$. Sequences containing concepts can therefore be compared not only syntactically but also semantically and the outcome of this comparison is no longer $\{0, \infty\}$ but can assume intermediary values between 0 and ∞ according to the distance between semantic concepts used in the service announcement and request. Services are described as a set of required inputs I_S, a set of provided outputs O_S and a set of provided properties P_S. Service descriptions may also contain access related information A_S that is not used by the matching algorithm, $S = \{I_S, O_S, P_S, A_S\}$.

Each input $i_S \in I_S$ has a name n_{i_S} and is semantically annotated with the code of one or more concepts that individually represents the input. Each concept is specific to a

single ontology. The set of all concepts used to annotate an input is Γ_{i_S} and the set of all ontologies used for the annotations is called Ω_{i_S}. For each concept in Γ_{i_S} corresponds an ontology in Ω_{i_S}, or in other words there is a membership function $C : \Gamma_{i_S} \rightarrow \Omega_{i_S}$ that given a concept used to annotate an input parameter returns the ontology to which that concept belongs. Each input is thus defined as $i_S = \{n_{i_S}, \Gamma_{i_S}, \Omega_{i_S}\}$. Similarly, outputs $o_S \in O_S$ and properties $p_S \in P_S$ are defined as $o_S = \{n_{o_S}, \Gamma_{o_S}, \Omega_{o_S}\}$ and $p_S = \{n_{p_S}, \Gamma_{p_S}, \Omega_{p_S}\}$. The semantic matching function is defined then as $\mathcal{M}_{sem} : \mathcal{S} \times \mathcal{R} \rightarrow D$ with $D = \mathbb{N} \cup \{\infty\}$, and it assigns *degrees* of match between a service announcement and a service request, where 0 means that the service and the request match exactly and ∞ means that there is no match. The degree of match between two descriptions is the sum of the degrees of match between each parameter of the discovery request and the service announcement. An announcement and a request may match with different degrees, depending on how the matching algorithm compares inputs, outputs and properties. In that case, the function $\mathcal{M}_{sem}$ returns the minimum degree of match among all possible match cases.

Based on the definitions above, we can specify a hybrid service discovery protocol that accepts syntactic service descriptions, semantic service descriptions or mixed service descriptions where some of the elements are semantically annotated and others are not. This protocol favors semantic matching whenever possible because we understand that semantic annotations augment the meaning of a description, and it only performs a syntactic matching when the semantic matching fails. However, semantic matching can fail for two different reasons: either because (i) the concepts used to annotate the request and the announcement belong to the same ontologies but are not related or (ii) the ontologies used to annotate the request and the announcement are different and the concepts cannot be compared. While in the first case the failure of semantic matching means that request and announcement certainly do not match, in the second case failure of semantic matching means that the request and the announcement are incomparable. As a consequence, in the first case syntactic matching is not performed because the more meaningful semantic match already determines that descriptions are incompatible, while in the second case syntactic matching is performed as an effort to solve the incompatibility of semantic annotations.

For a hybrid service discovery protocol, then, the semantic matching function $\mathcal{M}_{sem}$ must be redefined as $\mathcal{M}_{sem} : \mathcal{S} \times \mathcal{R} \rightarrow D$ with $D = \mathbb{N} \cup \{\infty, \perp\}$, where ∞ means that two concepts are unrelated and $\perp$ means that two concepts are incomparable. Since the final result of a semantic match is the sum of the semantic distance between each concept on requests and announcements, we define sum operations involving the incomparable ($\perp$) symbol as:

$$\begin{cases} i + \perp = \perp & \forall i \in \mathbb{N}; \\ \infty + \perp = \perp \\ \perp + \perp = \perp \end{cases}$$

When more than a possible semantic distance is possible between two concepts, the semantic matching chooses the smallest possible result. We define thus the following ordering when the incomparable symbol is involved: $\forall i \in \mathbb{N}, i < \infty < \perp$. The matching function

for this hybrid protocol, $\mathcal{M}_{hyb}$ is defined as $\mathcal{M}_{sem}$ when request and announcement elements are semantic annotated and the result is smaller than $\perp$, or $\mathcal{M}_{syn}$ otherwise. This protocol for hybrid service discovery is further detailed in [BRUC08].

4.2 Service Discovery Impact on Privacy

Service discovery data is privacy-sensitive because it represents the intentions of individuals publishing and consuming services in a service-oriented architecture. Even though a service discovery request does not exactly represent a service the user accesses (the user can look for a service without necessarily accessing it), it represents nonetheless a user intention. Service announcements, on the other hand, represent a service effectively offered by a provider, and can be used to infer personal information such as location and preferences.

Consider for instance a user looking for a drugstore around his current location. This service request reveals the user location and that he is probably sick. If the service request additionally specifies that the drugstore must have a specific type of medicine, more details about the user illness can be discovered. Similarly, a user that announces a death metal radio service is also disclosing his musical preferences.

Pervasive computing environments aggravate these risks for two reasons. First, people use pervasive computing systems everyday to help them in their daily activities and thus a pervasive service discovery protocol can potentially accumulate information about a great number of actions performed by many individuals. Second, users of service-oriented pervasive computing architectures play the roles of both clients (when consuming services available on the environment) and service providers (when offering to other users services running in their mobile devices) and as a result user private information is disclosed not only through service discovery requests but can also be deduced from service announcements.

When discovery related information such as service announcements and requests are stored across time and thoroughly analyzed, it is relatively easy for malicious service directories to infer user activities and personal preferences from apparently harmless data such as service names, request time, and context (e.g., location or network bandwidth). Recently, journalists were able to identify a woman based only on her anonymized Internet search history [BJ06]. Service discovery protocols are the SOA equivalent of Internet search engines, and the popularization of pervasive computing systems can multiply the occurrence of similar incidents.

The sensitive information disclosed during service discovery is a valuable asset for privacy intruders. An intruder can use the privacy invasion strategies defined in Chapter 2 to obtain personal data and build a privacy attack, which consists in gathering personal information to scrutinize a specific aspect of the individual's private life. In the next section we motivate and define the most important privacy attacks that an intruder can perform using service discovery data in pervasive computing.

4.2.1 Attack model

As defined in our trust model described in Chapter 1, we assume that user devices can authenticate the middleware code (for instance through a cryptographic signature) and not execute code that was maliciously modified. As a consequence, every node correctly follows the service discovery protocol and do not present Byzantine behavior. Service directories correctly execute the matching algorithm and include in the result every service they are supposed to include. However, we assume that intermediary nodes routing service discovery packets can save a copy of those packets for later analysis. We also assume that service directories can save the history of service announcements and requests and examine them later.

Service discovery data contains two types of information that can be used to invade privacy. One is related to user characteristics such as age, location or name. This data can be added to service discovery requests to find more relevant services, such as services physically close to the user. This issue is not specific to service discovery: whenever a client discloses personal data to interact with a service, that data reveals some private details to the service provider. To avoid this, users can use the techniques described in Section 2.2.3 to reduce private information disclosure.

The other type of information concerns user activities. Even if service discovery requests do not contain any user characteristic, they contain descriptions of services the user wants to access and will probably access. This information can reveal user interests and activities when stored and analyzed. Contrarily to the previous issue, this threat to user privacy is specific to service discovery because it is inherent to the protocol: to find a service, users must specify and send service descriptions.

Attackers that are interested in investigating user activities can use service discovery data for two purposes: either to keep track of all user activities on a certain period of time, or to learn if an individual used a specific service in a certain time interval. We are interested, thus, in avoiding the following privacy attacks:

1. **Profiling:** An attacker may be interested in monitoring all activities of a specific user during a period of time. To perform this attack, the attacker must choose a specific user, accumulate a number of service discovery requests from that user and analyze the services that the user was looking for on each request. The attacker may also be interested in identifying all services published by a user. The attack works similarly: the attacker chooses a specific user, accumulates all service descriptions he advertises and analyzes their contents.

2. **Chosen service monitoring:** An attacker may want to know if a user accessed a service. To perform this attack, an attacker must choose a specific user and a particular service, accumulate a number of service discovery requests from that user and search the particular service on each request. The attacker may also be interested to know if a user published a service. The attack works similarly: the attacker chooses a specific user and a particular service, accumulates a number of service descriptions that the user advertises, and look for that specific service on each announcement.

The two attacks described above can be performed by service directories handling discovery requests and announcements, by attackers that have reading access to the network (for instance by "sniffing" the network) or by nodes responsible for routing service discovery messages. In this chapter, we propose two protocols for service discovery, one that provides protection only against the first type of attack and that have lighter infrastructure requirements and another that protects users against both types of attacks but that demands stronger infrastructure requirements.

Before introducing our protocol, we present two naive approaches for increasing privacy of service discovery. By discussing how these approaches fail to address the problems discussed in this chapter, we intend to show the issues involved with designing a privacy-enhanced service discovery protocol and to motivate our solution.

4.2.2 Naive Protocols for Privacy-enhanced Service Discovery

The most straightforward solution to improve privacy during service discovery is to use symmetric key encryption to protect service announcements. Assuming that a client and a service provider share a key, the service provider encrypts the announcement using the shared key, and publishes the encrypted announcement in the directory. Clients encrypt the service discovery request using the same key and the service directory only has to test if both encrypted texts match. Figure 4.2 depicts this first protocol. Since service directories handle only encrypted versions of requests and announcements, identification of a published or requested service would be as hard as breaking a symmetric key encryption algorithm, which is a computationally hard problem.

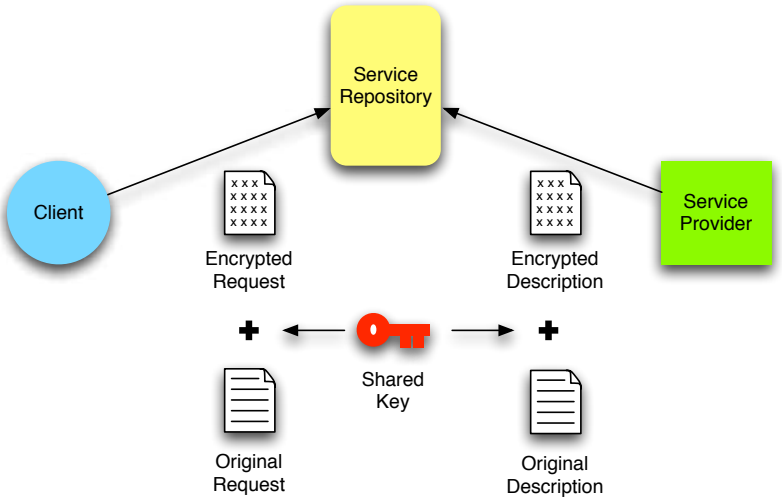


Figure 4.2: First Version of a Naive Protocol for Privacy-Aware Service Discovery

This protocol has many drawbacks. First of all, it only works if the service request and the service announcement are exactly the same and generate the same ciphertext after

encryption. However, this is very difficult in of open environments such as HMANETs. Services and clients are independently developed by different teams and are assumed to be loosely coupled. Existing services in the environment may require less input parameters than those provided by the user or may generate more outputs than required by the user, but still be useful to realize the user task. This first discovery protocol finds services only if the service request and the service announcement contain the same number of inputs, outputs and properties and they are syntactically identical.

This initial naive protocol can be improved to remove these limitations. In this second version, instead of encrypting the whole announcement, the service provider individually encrypts each parameter used to describe the service, i.e. its inputs, outputs and properties. Access-related information such as the service location and the message format can still be encrypted as a single block. The client also individually encrypts the parameters used to describe the service in the request. The service directory, then, can compare each input, output and property in the request with the service provider's equivalent. A match is found even if the service requires only a subset of the inputs provided by the client or if the client requires only a subset of the outputs or properties provided by the service. Figure 4.3 shows the improved protocol. Here again the directory only handles encrypted data, and identification of a service continues to be computationally hard.

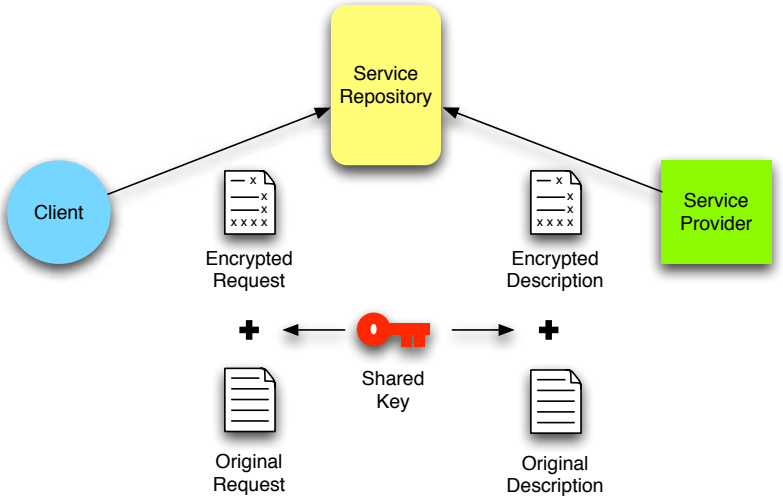


Figure 4.3: Second Version of a Naive Protocol for Privacy-Aware Service Discovery

This protocol only works when the inputs, outputs and properties defined in discovery requests are exactly the same as in service announcements. Moreover, it is effective only when a client wants to restrict service discovery to the services provided by a single provider. The client must share a different key with each service provider; if two different providers share the same key with the client they can decrypt the announcements of each other because the key is symmetric. As a result, a discovery request encrypted with a certain key only matches service announcements encrypted with the same key by a spe-

cific provider. To find a certain service regardless of service provider, a client would have to issue as many requests as keys he shares with service providers, which is inefficient particularly in open pervasive environments.

It is important to note that replacing symmetric encryption for asymmetric encryption does not solve the above issues. Either a user shares a pair of public and private key with each different service provider, and then he has to encrypt his request multiple times with different keys to find services from various service providers, or the user has a single pair of private and public keys and then the service provider has to publish different versions of the same announcement encrypted with the public key of each client that has access to the service. Also in this case a dictionary attack can be easily created to perform the chosen service monitoring attack described in Section 4.2.1. The attacker can store service discovery traffic from a specific user and since the protocol uses descriptions encrypted with public keys, the attacker can create the service description he wants to monitor, encrypt it with the public key, and compare this ciphertext with other ciphertexts in the client's service discovery history.

In the next sections we describe two protocols for privacy-enhanced service discovery that prevent the attacks described in Section 4.2.1 without the shortcomings of the naive protocols presented in this section. The first protocol supports privacy-enhanced syntactic and semantic matching of service announcements and protects the user against profiling attacks, while the second protocol also supports semantic and syntactic service matching but protects the user against both attacks of our attack model. We first define a model and introduce a notation to describe service discovery protocols in general and then, based on that model, we introduce our two privacy-enhanced protocols.

4.3 EVEY: A Protocol for Privacy-Enhanced Service Discovery

In this section we describe two protocols for privacy-enhanced service discovery, EVEY Light and EVEY Full, which present different privacy protection properties and characteristics. Each protocol considers a different attack model: EVEY Light protects users only against profiling attacks, as defined in Section 4.2.1, while EVEY Full protects users against profiling attacks but also against chosen service monitoring attacks. The main difference between both protocols concerns their infrastructure requirements and their efficiency in terms of network usage. While the first protocol requires less infrastructure and is more efficient, the second offers more protection at the cost of an overhead on message transmission and additional infrastructure requirements.

As we mentioned in Section 4.1.1, a service discovery protocol for pervasive computing must support both semantic and syntactic service descriptions because legacy devices or those with very limited resources may implement only a syntactic service discovery protocol, even if it is less effective. Both versions of EVEY support syntactic as well as semantic service descriptions.

A service discovery protocol must also be compatible with legacy service descriptions. This helps adoption of the new protocol since descriptions using other service discovery

protocols are also included in the results. For this reason, a privacy-enhanced protocols is more appropriate since it can still support regular service descriptions while clients and service providers requiring more privacy for service discovery can use the optional privacy-enhancement mechanism to increase privacy protection. The regular service matcher can also incorporate an optional plug-in to handle privacy-enhanced requests and announcements while still supporting regular descriptions.

4.3.1 EVEY Light

The goal of this first protocol is to propose an efficient solution for privacy-enhanced service discovery that protects users against profiling attacks and has lightweight infrastructure requirements. Efficiency in this case is measured in terms of the number and the size of exchanged messages between clients and service directories, and service providers and service directories. We first describe how the protocol handles syntactic service descriptions and after that how the protocol deals with semantic and hybrid service descriptions. After presenting the main ideas behind the protocol, we detail the processes of service publication, location, matching and selection in EVEY Light.

EVEY Light Overview

The main idea is to use encoded information from the original description to perform the match between a service request and a service announcement, and only encrypt service announcement data that is not used to verify a match. For instance, if WSDL [W3C01] is used as the language for service announcements and requests, the `binding` and the `service` sections contain useful information for accessing the service, but that is not used for matching requests and announcements. On the other hand, the `types`, `message` and `operation` sections define the service inputs, outputs and properties, which are the features specified by clients to discover a service.

If a service announcement is syntactic, the service provider transforms it into a privacy-enhanced announcement by replacing the service inputs, outputs and properties by their hashes, using a collision-free one-way hash function [Sch96] such as SHA-1 or MD5. Collision-free one-way hash functions provide two important properties. Assume that h is such a hash function, a is a plaintext and $h(a)$ is the resulting text after applying the hash function h to a.

One-way: given $h(a)$, it is computationally difficult to discover a

Collision-free: given a and $h(a)$, it is computationally difficult to find b such that $b \neq a$ and $h(b) = h(a)$

Service data not used for matching is encrypted with a key created from the data used for matching. Based on the required inputs defined in the service description, the service provider derives a symmetric key that is used to encrypt all service announcement data

not used for matching. The protocol uses service inputs to generate the key because this information is shared by clients and service providers: a client must be able to provide all inputs the service requires. This is not the case with outputs and properties because the client may be interested in a subset of the provided outputs and properties and the service provider does not know in advance which subset this might be. The privacy-enhanced announcement, thus, includes the hash of the service inputs, outputs and properties, as well as all access related data encrypted with a key derived from the service required inputs.

The client generates a privacy-enhanced syntactic discovery request by hashing provided inputs and required outputs and properties with the same hash function used by the service provider. The hash function, hence, should be agreed upon before protocol execution. However, under our attack model for EVEY Light (which consists only of profiling) this hash function does not need to be a secret. The service directory, then, compares the hashes on service descriptions and on service announcements to find a match. Since the hash function is collision-free, the probability that different requested and advertised features generate the same hash is negligible, and thus a match on hashed descriptions means that with overwhelming probability there is also a match on the original descriptions.

If there is one or more matches, the service directory sends the results to the client. The client receives the encrypted announcements and computes a decryption key based on the service inputs to recover the details required to access the service. Even though the set of inputs provided in the client discovery request contains the set of inputs in the service announcement, the client can determine the exact inputs the service uses by their hashes. Assuming that the service provider used the inputs according to a pre-defined order to generate the key (e.g., lexically or according to the order they appear in the description), the client is able to repeat the process and compute the symmetric key.

An attacker that has access to all traffic flowing from the client to the service directory cannot profile the user. Service requests contain only the hash of the service parameters sought by the user. Since the hash function is one-way, given the hash values it is computationally difficult for the attacker to discover the original parameters that generated them. Service descriptions contain the same hashed values but also other encrypted service information that could be useful to infer the service functionality. However, this information is encrypted with a key derived from the original service parameters. Since the attacker cannot discover those parameters from their hashes, if the key has enough entropy, the attacker cannot access the encrypted data neither.

Note, however, that an attacker can still perform the second attack described in Section 4.2.1. Suppose that the attacker is interested in knowing which clients are using a specific service. The attacker initially computes the hashes of service inputs, outputs and properties (the attacker can perform this step because the hash function used by the protocol is public). After that, the attacker captures discovery requests and simply searches for those hashes in the service descriptions. Following these steps, the attacker is able to identify which clients are looking for that service or which providers are announcing that service.

The semantic version of the protocol works similarly. A semantic service description

contains not only the names of inputs, outputs and properties but is also annotated with the concepts and ontologies that give more meaning to those names, as described in Section 4.1.2. In order to increase efficiency of the semantic matching algorithm, ontology reasoning is performed offline to infer all additional relations and concepts are encoded as described by [BKGI06]. The root concept in the ontology – usually Thing – is encoded as a prime number (e.g., 2) and its descendants are encoded as the product of the root's code by another prime number (e.g, 2x3=6, 2x5=10) and so on. This encoding enables the service matcher to rapidly identify if two concepts are related, it suffices to check if the smaller code divides the bigger. Hierarchy encoding is a recurring problem in computer science and many algorithms were proposed to optimize different aspects of the encoding process such as code size [KVH97, ABJ89] or the complexity of encoding nodes added to an already encoded hierarchy [AKBLN89, CF03].

A concept is unambiguously defined by its code and its ontology name. However, the same code may appear in different ontologies depending on their structure. The service provider transforms an encoded semantic service announcement into a privacy-enhanced service announcement by keeping the concept codes in plaintext and hashing the names of each ontology used to annotate the description. Other service data related to the service access is encrypted with a key derived from the IDs of the ontologies containing the concepts used to annotate the required service inputs.

Privacy-enhanced discovery requests contain the concept codes for provided inputs and required outputs and properties, as well as the hashes of each concept ontology ID. The service directory initially compares the hashes of the ontologies. If they match, the directory verifies if the concepts of the request are related to the concepts of the announcement by checking their codes. Positive matching results consist in requests and announcements possessing the same hashes of ontology IDs and respective concept codes that are related. Since identical hashes mean that with high probability the ontologies used by the request and by the announcement are the same, and the codes contained in the announcement are related to those in the request, the client can be sure that all announcements contained in the directory reply are relevant. After receiving the results, the client can decrypt service access information by deriving a key from the ontology IDs of the required inputs in the pre-defined order used by the service provider.

This protocol does not require any additional service in the infrastructure because there is no need for key distribution. Clients and service providers do not have to agree on an encryption key beforehand, nor they need to have access to a Public Key Infrastructure to obtain key certificates. This characteristic makes the protocol particularly appealing for use in exclusively ad hoc environments where nodes never have access (or cannot be assumed to have access) to a fixed infrastructure. The protocol is also efficient in terms of network usage because a single discovery request from the client is enough to find all compatible services in the directory, and the directory reply contains only services that are relevant to the client's request.

Clients and service providers must perform additional hashing and symmetric encryption operations, which are not needed in regular service discovery protocols. This overhead remains nevertheless acceptable given the protocol benefits. It is important to notice as

well that EVEY Light causes no additional overhead for service directories, since they do not have to perform any hash or encryption operation and just compare strings or check if semantic concepts are related as in regular matching algorithms.

Description of EVEY Light

Our first proposed privacy-enhanced service discovery protocol introduces the following elements in the general service discovery model introduced in Section 4.1.3:

- A language for privacy-enhanced service announcements $\mathcal{S}_{pe} = \{\mathcal{A}_{\mathcal{S}_{pe}}, \mathcal{F}_{\mathcal{S}_{pe}}\}$ and two functions $\mathcal{P}_S : \mathcal{S} \times K \rightarrow \mathcal{S}_{pe}$ and $\mathcal{P}_S^{-1} : \mathcal{S}_{pe} \times K \rightarrow \mathcal{S}$;
- A language for privacy-enhanced service requests $\mathcal{R}_{pe} = \{\mathcal{A}_{R_{pe}}, \mathcal{F}_{R_{pe}}\}$ and a converting function $\mathcal{P}_R : \mathcal{R} \rightarrow \mathcal{R}_{pe}$;

In the above definitions, K is a set of symmetric cryptographic keys, which means that the same key is used by the provider to generate a privacy-enhanced announcement from a regular announcement using $\mathcal{P}_S$ and by the client to recover the original service announcement from the privacy-enhanced version using $\mathcal{P}_S^{-1}$. We assume that a client can differentiate between a plaintext obtained by decrypting a ciphertext with the correct key from a plaintext obtained by decrypting a ciphertext using the wrong key. This is trivial in practice because service descriptions follow a well-defined format (WSDL for instance), and the probability that a ciphertext decrypted with a wrong key will generate a plaintext that is different from the original plaintext but respects this format is very small.

The matching function used by this protocol is the same function $\mathcal{M}_{hyb}$ defined in Section 4.1.3, and it computes the degree of match between a request and an announcement as their semantic distance if both are semantic annotated and the result is smaller than $\perp$ or the syntactic distance otherwise.

Service Publication: Service publication is the process of creating a service announcement and making it available to clients. Service providers use the function $\mathcal{P}_S$ on the original announcement S to obtain the privacy-enhanced version S_{pe}. As outlined in Section 4.3.1, the service provider replaces a syntactic element of a service announcement by its hash and the identification of each ontology used to annotate the description by its hash. Besides that, all service access information is encrypted with a symmetric key derived from service description data.

A privacy-enhanced service announcement is defined as $S_{pe} = \{I_{S_{pe}}, O_{S_{pe}}, P_{S_{pe}}, A_{S_{pe}}\}$ where for each input $i_{S_{pe}} \in I_{S_{pe}}$, $i_{S_{pe}} = \{h(n_{i_S}), \Gamma_{i_S}, h(\Omega_{i_S})\}$, with outputs and properties defined likewise. Here, $h(\Omega_{i_S})$ means that each member of the set Ω_{i_S} is individually hashed using function $h()$. Privacy-enhanced service access information consists of the original access information encrypted, $A_{S_{pe}} = \{A_S\}_k$.

The service provider generates the encryption key k using information contained in the original service announcement S. If the announcement is fully syntactic, the names of

the required inputs are concatenated and hashed using MD5 to generate a 128-bit key . If the announcement is fully semantic, the IDs of the ontologies used to annotate the service required inputs are concatenated and used as the input for MD5. If the announcement is hybrid, the names of syntactic input elements are concatenated to the IDs of the ontologies used to annotate the semantic input elements and used as the input for MD5. The order in which these parameters are concatenated must follow a pre-defined order commonly agreed among clients and service providers (e.g., lexically).

Malicious third-parties are only able to read the codes of the concepts used to semantically annotate inputs, outputs and properties (the sets Γ_i, Γ_o and Γ_p), the hashes of the ontologies used to annotate these elements (sets $h(\Omega_i)$, $h(\Omega_o)$ and $h(\Omega_p)$), the hashes of inputs, outputs and properties names ($h(n_i)$, $h(n_o)$ and $h(n_p)$), and the encrypted service access data ($\{A_S\}_k$). This information is not sufficient to discover the original description of the service. In Section 4.3.3 we present a more detailed discussion of the privacy properties provided by EVEY Light.

Service Location: To locate a service, clients send discovery requests to service directories where they are used as an input to the matching algorithm to find appropriate services among the available announcements. Discovery requests do not contain any access related information, and thus are defined as $R = \{I_R, O_R, P_R\}$. Clients use the function $\mathcal{P}_R$ on the original request R to generate its privacy-enhanced version R_{pe}. Similarly to the generation of privacy-enhanced announcements, a syntactic element of a discovery request is replaced by its hash while for a semantic element the IDs of the ontologies used to annotate it are hashed. A privacy-enhanced service request is then defined as $R_{pe} = \{I_{R_{pe}}, O_{R_{pe}}, P_{R_{pe}}\}$ where for each input $i_{R_{pe}} \in I_{R_{pe}}$, $i_{R_{pe}} = \{h(n_{i_R}), \Gamma_{i_R}, h(\Omega_{i_R})\}$ with outputs and properties defined likewise.

Service Matching: The algorithm for service matching supports syntactic matching, semantic matching or a combination of both. For the reasons explained in Section 4.1.3, the algorithm favors semantic matching of service announcements and requests, and only performs a syntactic match if a given input, output or property of a service announcement or request is not semantically annotated, or if the annotations in announcements and requests use different ontologies and are incompatible. The algorithm requires two functions to compute the degree of match between a privacy-enhanced service announcement S_{pe} and a privacy-enhanced service request R_{pe}, one that computes the *syntactic distance* (d_{syn}) and another that computes the *semantic distance* (d_{sem}) between two elements. The syntactic distance between a service announcement input $i_{S_{pe}} = \{h(n_{i_S}), \Gamma_{i_S}, h(\Omega_{i_S})\}$ and a service request input $i_{R_{pe}} = \{h(n_{i_R}), \Gamma_{i_R}, h(\Omega_{i_R})\}$ is defined as:

$$d_{syn}(i_{S_{pe}}, i_{R_{pe}}) = \begin{cases} 0 & \text{if } h(n_{i_S}) = h(n_{i_R}); \\ \infty & \text{otherwise.} \end{cases}$$

This function is similarly defined for outputs and properties. The semantic distance between a service announcement input $i_{S_{pe}}$ and a service request input $i_{R_{pe}}$ is the minimum of all semantic distances between concepts used to annotate $i_{S_{pe}}$ and $i_{R_{pe}}$. Since each

input may be annotated with more than one concept, the minimal distance identifies the concepts from the request and the announcement that are closest in meaning and hence form a potentially more relevant match.

Two concepts belonging to the same ontology can be related (if they belong to the same subtree) or not (if they belong to different subtrees). The semantic distance between concepts $\gamma_{i_S} \in \Gamma_{i_S}$ and $\gamma_{i_R} \in \Gamma_{i_R}$ is:

$$d_{sem}(\gamma_{i_S}, \gamma_{i_R}) = \begin{cases} |l(\gamma_{i_S}) - l(\gamma_{i_R})| & \text{if } h(\omega_{i_S}) = h(\omega_{i_R}) \text{ and } \gamma_{i_S} \text{ is related to } \gamma_{i_R}; \\ \infty & \text{if } h(\omega_{i_S}) = h(\omega_{i_R}) \text{ and } \gamma_{i_S} \text{ is not related to } \gamma_{i_R}; \\ \perp & \text{otherwise.} \end{cases}$$

In this definition, $l(\gamma)$ represents the level of the concept γ in its ontology hierarchy. The semantic distance between concepts used to annotate outputs and properties is defined similarly. Matches between the concepts used to annotate the request and the announcement may result in different distances. For instance, if the client and the service provider annotate an input using concepts from two different ontologies, the distance between the concepts in one ontology may be different from the distance in another ontology. For this reason, the final semantic distance is the minimal among all possible distances between two concepts. The hybrid distance d between two inputs $i_{S_{pe}}$ and $i_{R_{pe}}$ is the semantic distance between them if both are semantic annotated and their distance is smaller than $\perp$ or the syntactic distance otherwise, similarly for outputs and properties.

To verify if a service announcement matches a service request, the matching algorithm uses the hybrid distance to compute all possible distances between inputs, outputs and properties independently. Required inputs on service announcements are compared to provided inputs on service requests, while required outputs and properties on service requests are compared to provided outputs and properties on service announcements. The final match result is the the sum of the minimum distances between inputs, outputs and properties. If one input, output or property does not match, one of the sum terms will be infinity and the total sum result will also be infinity.

$$\mathcal{M}_{pa}(S_{pe}, R_{pe}) = d(I_{S_{pe}}, I_{R_{pe}}) + d(O_{R_{pe}}, O_{S_{pe}}) + d(P_{R_{pe}}, P_{S_{pe}})$$

The matching algorithm checks all services stored in the directory for a match. Whenever $\mathcal{M}_{pa}(S_{pe}, R_{pe}) < \infty$ we say that the announcement and the request match. To decide if an announcement matches a request, the service directory does not need to access encrypted information, nor the plaintext that originated the hashes contained in the descriptions. All matched privacy-enhanced service announcements are returned to the client ordered from the smallest matching distance to the biggest, with fully semantic matches before hybrid matches and syntactic matches at the end. Since we use a collision-free hash function, there is a extremely high probability that all results returned by the matching algorithm are relevant to the client query. In other words, the probability that this matching algorithm returns a false positive is the same as the probability that two different texts hash to the same result, which is about 2^{-63} for SHA-1 today[1] (assuming

[1] www.schneier.com/blog/archives/2005/08/new_cryptanalyt.html

that an attacker wants to intentionally forge a fake service description that will generate a false positive). More details about the privacy properties of the protocol in Section 4.3.3.

Service Selection: As explained above, all discovery results that the client receives are relevant. The client then may select the best result according to a client-defined criteria, for instance the one with the smallest semantic distance from the request or the one that requires the smallest number of inputs to minimize information disclosure. To finally access the service, however, the client must first decrypt the access related information $\{A_S\}_k$ of the announcement.

According to the type of match found by the service directory, the client knows how to compute the decryption key k: it is generated from the names of syntactic input parameters (n_{iS}) or the ontology IDs of semantic annotated input parameters (Ω_{iS}). As the key generation algorithm uses a pre-defined ordering to generate a key based on this data, the client can concatenate the names of input parameters or the names of input ontologies in that same order (e.g., lexically) to generate the symmetric key for decryption.

4.3.2 EVEY Full

The second protocol intends to protect users from both attacks described in Section 4.2.1, profiling and chosen service monitoring. Similarly to the first protocol, this second protocol also uses encoded information to perform a match between service announcements and requests. In EVEY Full, however, service descriptions are generalized to represent a set of services instead of a single service, complicating analysis of request and announcement content. We start by describing how service descriptions are generalized and after we discuss some infrastructure requirements of the protocol. After presenting an overview of EVEY Full, we detail how the protocol handles service publication, location, matching and selection.

EVEY Full Overview

It is important to note that protection against chosen service monitoring requires that authorized clients and service providers share a secret. Suppose that clients and service providers do not share a secret. Any client looking for a service can generate a request that matches the service announcement created by the service provider. In particular, a malicious service directory that wants to monitor which clients are interested in accessing a specific service can generate a request for that service, check which announcements match that service and keep track of all users that receive that service as a discovery result. If authorized clients and service providers do not share a secret, it is impossible to distinguish between a malicious client monitoring discovery requests or a regular client looking for a service. Clients must know something that the attacker does not know.

Similarly to the first protocol, this second protocol also uses encoded information to perform a match between service announcements and requests, while other data not

required during match is encrypted. Since matching data is not encrypted, clients do not need to encrypt the request with different keys and can issue a single request to find services offered by different providers. However, if this unencrypted information can uniquely identify a service, an attacker can still perform the chosen service monitoring attack defined in Section 4.2.1. The attacker must only choose a service and check the unencrypted matching data contained in discovery requests or announcements from a user to learn if he looked for or published that service.

For this reason, the matching algorithm uses encoded data that is ambiguous, or in other words, that do not uniquely identify a service. Client requests contain a generalized service description that matches not only the service sought by the client but also possibly other services. Service announcements also contain generalized descriptions that characterize the service being offered but can also describe other services potentially unrelated. A matching between generalized service descriptions means that there is a high probability that the original service request and the original service announcement match. Conversely, there is a small probability that the match is a false positive: even though the generalized versions match, the original request and announcement are different but produce the same generalized description.

Syntactic service descriptions are generalized using a hash function with weak collision resistance. A hash function with weak collision resistance $h(x)$ has the property that it is relatively easy to find distinct values x and y such that $h(x) = h(y)$. For instance, instead of containing the real category name (e.g., *carDealer*) client requests contain the hash of the category name (e.g., $h($"carDealer"$)$). Announcements matching this request would consist of services in the *carDealer* category but also of other services whose categories, when hashed, collide with $h($"carDealer"$)$. Syntactic inputs, outputs and properties are replaced by their weak hash, and the whole original description is encrypted using a key shared between the service provider and clients authorized to discover that service.

Semantic service descriptions contain elements that are semantically annotated with concepts belonging to specific ontologies, increasing the description expressiveness. We assume that semantic service descriptions use the encoding technique proposed in [BKGI06] and described in Section 4.3.1. To better explain our approach for generalizing semantic annotated service descriptions, consider Figure 4.4 that shows a snippet of two ontologies (I) and (II) after classification and their respective encoding (III) and (IV) using the encoding technique of [BKGI06]. We assume that ontology classification is deterministic and the order of child nodes is defined lexically.

If we only take into account the encoded versions of the ontologies, we notice that different ontologies with the same structure generate identical encoded hierarchies. As a consequence, if a semantic annotation contains the code *66* and does not mention the ontology name, one cannot tell if it is related to "breast cancer" or "Leninism". Service announcements annotated with encoded ontology concepts – and not containing each concept's ontology name – no longer uniquely represent a single service but instead a set of heterogeneous services. Our protocol generalizes semantic annotated service descriptions by replacing the original annotations with encoded concepts and removing the ontology identification of each concept. The original semantic description is also encrypted using a

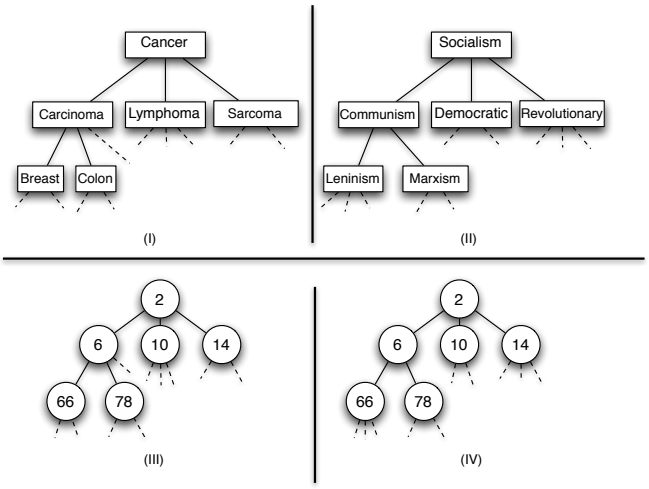


Figure 4.4: Possible Ontology Encoding

key shared between the service provider and clients allowed to discover the service.

Based only on privacy-enhanced versions of requests and announcements, the matching algorithm is able to find potential matches of syntactic, partially semantic annotated and fully semantic annotated descriptions. As the same privacy-enhanced description may represent different services, the service directory cannot precisely determine which service a client is searching or a service provider is offering. If the matching algorithm finds multiple matches, it orders results according to their matching distance to the request, with semantic results before syntactic results. Clients must review service discovery results to identify possible false positives. Service descriptions are accompanied by a key description document that details the key that the service provider used to encrypt the description, and for each result the client must verify if it possesses the key to decrypt the complete service description. After that, the client uses the key to obtain the original service description and checks if it is a real match or a false positive.

The protocol uses cryptography to protect service descriptions against chosen service monitoring attacks. However, key management in ad hoc environments is a complex issue since key distribution services or certification authorities may not be available when needed. The requirement that clients and service providers share a key, thus, may seem too strong for pervasive computing environments and must be discussed in more details to evaluate its feasibility.

First of all, it is important to notice that privacy-enhanced service descriptions are not a replacement for service access control. The goal of encrypting service descriptions is to reduce the number of entities able to understand their contents, not to control which clients have access to the service. As such, privacy-enhanced descriptions may not be protected with the same level of detail as access control. In other words, even if only a

small group of users (e.g., a department) from a larger set (e.g., a company) is able to access a service, the service description may be readable by all members of the larger set even if they cannot access it. What is important is that clients not related to the service (e.g., other company's employees) are not able to understand the description.

This is not the equivalent of having a trusted service directory containing services from that company. Suppose that the company's employees meet on a restaurant. Using this protocol, they would be able to understand each other's service descriptions even if the service directory is an untrusted device from someone else in the same restaurant. Other customers would not be able to understand these descriptions, though, because they would not have the company's key. Obviously, in situations where the overhead is acceptable, service providers can still share different keys with each client and publish multiple versions of the same service description encrypted using each key individually.

We also assume an environment where multiple identity providers are responsible for authenticating users and managing their credentials. A single user may have one account on the identity provider of his company, another on the identity provider of his cellphone operator, another on the identity provider of his internet service provider (ISP) and yet another on the identity provider of a specific community to which he belongs. Service providers may be interested to offer services whose visibility is restricted to one or some of those communities, and can make use of identity providers to organize users in groups and to define respective group keys according to the service provider's needs. Service providers can use those group keys to encrypt service descriptions, and identity providers distribute those keys to users when they authenticate.

Finally, even though users can create ad hoc networks in pervasive computing and a substantial number of interactions occur in infrastructure-less environments, we assume that users will eventually have access to a fixed infrastructure, where they will be able to authenticate with an identity provider and obtain the keys they are entitled to. To avoid that a user who is no longer member of the group still accesses the content of a service description, service providers and identity providers can agree on how long a key remains valid to force users to re-authenticate and recover a fresh key.

Figure 4.5 shows our proposed model for identity management in pervasive computing. Service providers are represented by squares, clients by circles, identity providers by hexagons and the service directory is represented by a rounded-corner rectangle containing service announcements. A dotted line from a client to an identity provider means that the client has an account with the identity provider, while a dotted line from a service provider to an identity provider means that the service provider trusts the identity provider to manage keys related to the services it publishes. Finally, arrows represent service announcements from service providers to the service directory.

Users can have accounts on multiple identity providers according to the communities they belong to. In this picture, user X has an account with identity providers A and B, while user Y has an account with identity provider B. Service providers on the other hand have agreements with some identity providers and trust the assertions produced by them. In this case, service providers 1 and 2 trust identity provider A, while service provider 3 trusts identity providers A and B.

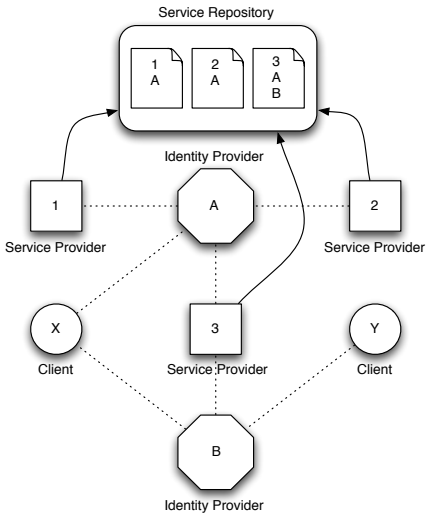


Figure 4.5: A Model for Pervasive Identity Management

Each service provider publishes a privacy-enhanced service description, along with a specification of the key used to encrypt the description. This specification contains details about the cryptographic protocol used but also the identity provider trusted to distribute the key. When receiving service discovery results, clients check if they possess an account on any of the identity providers listed in the description. If that is the case, the user can decrypt the original message and check if the result is a false positive or not. In Fig. 4.5, client Y can only decrypt the description published by service provider 3, while client X can decrypt service descriptions published by services 1, 2 and 3.

This approach for identity management has the same advantages as a Public Key Interface (PKI) with a certification authority (CA). It decouples credential administration from applications, thus improving flexibility. Users do not have to own accounts with each service provider of every service they want to use. Rather, users possess accounts with a certain number of identity providers depending on their activities and interests, and service providers rely on those identity providers to authenticate and distribute keys to users. It also does not require all entities to be online at the same time. Users can authenticate with identity providers and obtain keys before they receive service discovery results, service providers can establish relations with identity providers and publish service announcements when clients are offline, and clients can check discovery results and decrypt service descriptions when identity providers are offline, if they are already authenticated and in possession of decryption keys.

This model for identity management can be implemented using standards such as SAML [OAS05a]. SAML defines protocols, messages and formats for the exchange of authentication information between clients, service providers and identity providers. Using SAML, users can prove that they were authenticated by a given identity provider or produce an evidence signed by an identity provider that they possess some attribute.

SAML is based on Web technologies such as XML, HTTP and SOAP, which are already part of a Web Service oriented pervasive computing infrastructure.

In SAML, a profile defines constraints and extensions to SAML to support a given scenario. For instance, the Web Browser Single Sign-On profile defines the flow of messages, the types of messages and the message formats to support a scenario where a user wants to access a protected resource at a service provider and uses an assertion generated by an identity provider to prove that he possesses the credentials to access the resource [OAS05b]. Existing SAML profiles, however, were created with Web authentication scenarios in mind, and require that all parties are online to allow user access to a resource. To support the above scenario, a new SAML Profile is needed. This new profile must enable service providers to publish their `AuthnRequest` messages along with privacy-enhanced service descriptions. `AuthnRequest` messages contain the specification of the credentials that the client must present to have access to a resource. The profile must also specify a new message flow. Since SAML profiles are designed to web-based scenarios, they usually assume that client, identity provider and service provider are online at the same time. To support our use case, a message flow supporting disconnected interactions is necessary.

Description of EVEY Full

Our second proposed privacy-enhanced service discovery protocol introduces the following elements in the general service discovery model:

- A language for describing authentication requirements $\mathcal{A} = \{A_{Auth}, F_{Auth}\}$;
- A language for privacy-enhanced service announcements $\mathcal{S}_{pe} = \{A_{S_{pe}}, F_{S_{pe}}\}$ and two functions $\mathcal{P}_S : \mathcal{S} \times K \rightarrow \mathcal{S}_{pe}$ and $\mathcal{P}_S^{-1} : \mathcal{S}_{pe} \times K \rightarrow \mathcal{S}$;
- A language for privacy-enhanced service requests $\mathcal{R}_{pe} = \{A_{R_{pe}}, F_{R_{pe}}\}$ and a converting function $\mathcal{P}_R : \mathcal{R} \rightarrow \mathcal{R}_{pe}$;
- A privacy-enhanced matching function $\mathcal{M}_{pa} : \mathcal{S}_{pe} \times \mathcal{R}_{pe} \rightarrow D$, $D = \{\mathbb{N} \cup \infty\}^n$.

In the above definitions, K is a set of symmetric cryptographic keys such that the service provider owns the encryption key $k \in K$ and the client owns the decryption key $k^{-1} \in K$. The language $\mathcal{A}$ describes the identity provider that the client must use to obtain the decryption key and other details about the key such as cryptographic algorithm and freshness or about authentication requirements such as user attributes.

The matching function $\mathcal{M}_{pa}$ computes all possible degrees of match between a service announcement and a request. As we will see later, the matching algorithm of EVEY Full cannot identify when two annotations are incompatible, so the matching results of this algorithm do not contain the $\perp$ symbol as the algorithm defined in Section 4.1.3. The algorithm can still detect when a request and an announcement do not match, in which case the function result is $\{\infty\}$. It is important to note that if there is a match between *privacy-enhanced* versions of a service announcement and request, it does not necessarily

imply that there is a match between the *original* versions of a service announcement and request. In other words, given $S_{pe} = \mathcal{P}_S(S, k)$ and $R_{pe} = \mathcal{P}_R(R)$, $\mathcal{M}_{pa}(S_{pe}, R_{pe}) \neq \{\infty\} \not\Rightarrow \mathcal{M}_{sem}(S, R) < \infty$. The opposite, however, is true: if the privacy-enhanced versions do not match, then the original versions do not match neither, $\mathcal{M}_{pa}(S_{pe}, R_{pe}) = \{\infty\} \Rightarrow \mathcal{M}_{sem}(S, R) = \infty$. The matching output contains not only the computed distances but also the matching elements of the announcement and request that produced each distance, to allow clients to verify if the match is real or a false positive.

In the remaining of this section we describe how a service provider creates a privacy-enhanced service description using $\mathcal{P}_S$ and how a client creates a privacy-enhanced request using $\mathcal{P}_R$. After that we detail the matching process explaining how the privacy-enhanced matching algorithm $\mathcal{M}_{pa}$ performs privacy-enhanced syntactic, semantic and hybrid matches. Finally we explain how clients can identify if a match of privacy-enhanced descriptions is also a match of original descriptions.

Services are described as a set of required inputs I_S, a set of provided outputs O_S, a set of provided properties P_S and some access related information A_S, $S = \{I_S, O_S, P_S, A_S\}$. Each input $i_S \in I_S$ has a name n_{i_S} and may be semantically annotated with the code of one or more concepts that individually represent the input, and each concept belong to a specific ontology. The set of all codes used to annotate an input is Γ_{i_S} and the set of all ontologies used for the annotations is called Ω_{i_S}. For each concept in Γ_{i_S} corresponds an ontology in Ω_{i_S}, or in other words there is a membership function $C : \Gamma_{i_S} \rightarrow \Omega_{i_S}$ that given a concept used to annotate an input parameter returns the ontology to which that concept belongs. Each input is thus defined as $i_S = \{n_{i_S}, \Gamma_{i_S}, \Omega_{i_S}\}$. Similarly, outputs $o_S \in O_S$ and properties $p_S \in P_S$ are defined as $o_S = \{n_{o_S}, \Gamma_{o_S}, \Omega_{o_S}\}$ and $p_S = \{n_{p_S}, \Gamma_{p_S}, \Omega_{p_S}\}$.

Service Publication: Service publication is the process of creating a service announcement and making it available to relevant clients. Service providers use the function $\mathcal{P}_S$ on the original announcement S to obtain the privacy-enhanced version S_{pe}. As outlined in Section 4.3.2, a syntactic element of a service announcement is replaced by its weak hash while a semantic element has the set of ontologies used to annotate it omitted. The whole service description is also encrypted using key k.

A privacy-enhanced service announcement is then defined as $S_{pe} = \{I_{S_{pe}}, O_{S_{pe}}, P_{S_{pe}}, \{S\}_k\}$ where for each input $i_{S_{pe}} \in I_{S_{pe}}$, $i_{S_{pe}} = \{h(n_{i_S}), \Gamma_{i_S}\}$, with outputs and properties defined likewise. After creating the privacy-enhanced service announcement, the service provider sends it to the service directory, along with a specification of the key used to encrypt the announcement written in language $\mathcal{A}$.

Service Location: To locate a service, clients send service requests to service directories where they are used as an input to the matching algorithm to find appropriate services among the available announcements. Clients use the function $\mathcal{P}_R$ on the original request R to generate its privacy-enhanced version R_{pe}. A syntactic element of a service request is replaced by its weak hash while for a semantic element the set of ontologies used to annotate it is omitted. A privacy-enhanced service request is then defined as $R_{pe} = \{I_{R_{pe}}, O_{R_{pe}}, P_{R_{pe}}\}$ where for each input $i_{R_{pe}} \in I_{R_{pe}}$, $i_{R_{pe}} = \{h(n_{i_R}), \Gamma_{i_R}\}$ with outputs and

properties defined likewise.

Privacy-enhanced service requests do not contain encrypted information. The matching algorithm only uses generalized service descriptions to find a service announcement that potentially matches the request. As requests do not contain encrypted information, even if the key shared between a client and a service provider is compromised, an attacker is unable to identify the services that a client is looking for.

Service Matching: The algorithm for service matching supports syntactic matching, semantic matching or a combination of both. Traditional algorithms for matching hybrid service descriptions, as described in Section 4.1.3, favor semantic matches over syntactic matches because the former are more expressive than the later. Here, since the semantic match may be a false positive, the service directory must also perform a syntactic match.

The algorithm requires two functions to compute the degree of match between a privacy-enhanced service announcement S_{pe} and a privacy-enhanced service request R_{pe}, one that computes the *syntactic distance* (d_{syn}) and another that computes the *semantic distance* (d_{sem}) between two elements. The syntactic distance between a service announcement input $i_{S_{pe}} = \{h(n_{i_S}), \Gamma_{i_S}\}$ and a service request input $i_{R_{pe}} = \{h(n_{i_R}), \Gamma_{i_R}\}$ is defined as:

$$d_{syn}(i_{S_{pe}}, i_{R_{pe}}) = \begin{cases} 0 & \text{if } h(n_{i_S}) = h(n_{i_R}); \\ \infty & \text{otherwise.} \end{cases}$$

This function is similarly defined for outputs and properties. The semantic distance between a service announcement input $i_{S_{pe}}$ and a service request input $i_{R_{pe}}$ is the bag (multiset) containing all semantic distances between concepts used to annotate $i_{S_{pe}}$ and $i_{R_{pe}}$, where the semantic distance between concepts $\gamma_{i_S} \in \Gamma_{i_S}$ and $\gamma_{i_R} \in \Gamma_{i_R}$ is:

$$d_{sem}(\gamma_{i_S}, \gamma_{i_R}) = \begin{cases} |l(\gamma_{i_S}) - l(\gamma_{i_R})| & \text{if both are related;} \\ \infty & \text{otherwise.} \end{cases}$$

In this definition, $l(\gamma)$ represents the level of the concept γ in its ontology hierarchy. The semantic distance between concepts used to annotate outputs and properties is defined similarly. As the algorithm cannot compare ontology IDs (they are omitted in requests and encrypted in announcements), it cannot distinguish compatible and incompatible annotations. For this reason, semantic matching results do not contain the symbol $\perp$.

The result is a multiset because each parameter can be annotated with multiple concepts from different ontologies. As the algorithm compares the set of concepts used to annotate an input in the request and the set of concepts used to annotate an input in the announcement, it obtains different semantic distances. However, since the algorithm does not know which of those results is a false positive and which is not, it cannot send only the minimum matching distance as the original semantic matching algorithm. Instead, the final result set must admit all possible matching results including elements with the same value. The hybrid distance d between two inputs $i_{S_{pe}}$ and $i_{R_{pe}}$ is defined as the multiset containing all semantic distances between them and their syntactic distance.

Required inputs on service announcements are compared to provided inputs on service requests, while required outputs and properties on service requests are compared to provided outputs and properties on service announcements. As an input in the announcement is compared to the inputs in the request, the hybrid distance function generates many multisets containing the semantic matching results for each parameter and many values representing the syntactic match distance to each parameter. If at least one multiset of semantic results contain an element different from ∞ we say that a provided input potentially matches semantically the required input. If at least one of the syntactic match results is different from ∞, we say that a provided input potentially matches syntactically the required input. If any of the two cases is true, we say that a provided input potentially matches syntactically and semantically the required input, and similarly for outputs and properties.

A potential match between a privacy-enhanced request and a privacy-enhanced announcement happens when all required inputs potentially match some of the provided inputs, and all required outputs and properties potentially match some of the provided outputs and properties. If all matches are potentially semantic and potentially syntactic, we have a potential semantic and syntactic match between request and announcement. If some potential matches are semantic and others are syntactic, we have a potential hybrid match. Finally, if all potential matches are semantic (correspondingly syntactic), we have a potential semantic (correspondingly syntactic) match. All potentially matching privacy-enhanced service announcements are returned to the client along with their corresponding key specification files. The directory also includes on its reply the type of potential match that was found: if semantic and syntactic, hybrid, only semantic or only syntactic.

It is important to note that the semantics of this privacy-enhanced matching function is different than traditional matching. When a syntactic match occurs, it means that "service announcement S is syntactically equivalent to service request R", while a semantic match means that "the semantic distance between service announcement S and service request R is n". When a privacy-enhanced match happens, however, it means that "if S matches R, than it is a semantic match" for a potentially semantic match or "if S matches R, some elements of S match elements of R semantically and other elements of S match elements of R syntactically" for a potentially hybrid match.

To compute all possible distances between a privacy-enhanced request and a privacy-enhanced service announcement, the matching algorithm needs only to access information that is available in plain-text in both the announcement S_{pe} and the request R_{pe}, namely, the hashed name and concept codes for inputs, outputs and properties. This information does not allow the entity running the matching algorithm to identify the original service announcement S and request R that yielded the corresponding privacy-enhanced versions. The information used by the matching algorithm is not sensitive, and as a result the service directory running the matching algorithm no longer needs to be trusted simplifying replication and improving scalability of the service discovery protocol.

Service Selection: Clients receive discovery results composed of privacy-enhanced service announcements potentially matching the privacy-enhanced request, the type of potential

match found and key specification documents describing the type of key used to encrypt the original file and how the client can obtain it.

To check for false positives, the client must first eliminate the announcements it is unable to access and then filter out announcements that do not match the request and generated false matches. First the client checks the key specification of the first announcement in the list of results. The key specification contains the identity provider responsible for distributing the decryption key and further information such as which attributes the client must possess to have access to the key. The client checks if it has an account on that identity provider and if it already possesses the decryption key. If the client does not have an account on the identity provider it does not have access to that description and simply passes to the next discovery result.

If the client has an account on the specified identity provider, it may already be in possession of the key (if for instance it is already authenticated to that provider and the key obtained is still valid) or not. If the client does not have the key, it authenticates with the identity provider and recovers the corresponding decryption key. Having control of the decryption key, the client decrypts the original description contained in the announcement and verifies if the type of match signaled by the directory holds. For a semantic match, for instance, the client checks if the ontologies used to annotate the request are the same as the ones used to annotate the announcement. For a syntactic match, the client checks if the request parameters are syntactically equivalent to the announcement parameters. If the client finds an error, the announcement is a false positive and the client passes to the next result of the list.

Clients can choose the best filtering strategy according to their needs. For instance, the client can choose to decrypt all results provided by the directory to have a higher number of services to select from, or it can choose to decrypt announcements until it finds a suitable match. The client can also start by checking first potentially semantic and syntactic matches since those results have higher probability of providing a real match, or start by potentially semantic matches that, if confirmed, provide more meaningful results. The client may also try to check first the results encrypted with keys it already possesses, and only if those are not adequate the client authenticates with other identity providers and obtains additional decryption keys.

4.3.3 EVEY Assessment

Even though privacy enhancements, in general, cause an overhead because they require additional software layers to increase privacy, this overhead must minimally affect the user experience to favor protocol adoption. The straightforward protocol to protect privacy in service discovery – which consists in clients downloading all descriptions stored by service directories – provides high privacy but incurs an excessively high cost in terms of network traffic and computation. Our goal is to propose a service discovery protocol that generates less network traffic than the straightforward protocol but that still protects user privacy.

It is also important to evaluate the performance overhead caused by the protocol. Generally, privacy-enhancements require that additional resources are used to privacy protec-

tion. This overhead, however, must be minimized to encourage protocol usage. If privacy protection consumes a large quantity of resources, users may view privacy-enhancements more as an inconvenient than as a benefit. In the following sections we analyze the privacy protection provided by both versions of EVEY compared to the additional network traffic that the protocols generate, and also the performance overhead that the protocol produces when generating privacy-enhanced descriptions and when performing privacy-enhanced matches.

Privacy Protection Assessment of EVEY Light

In this section we analyze the probability that the matching algorithm from EVEY Light provides false positives in service discovery results. Ideally, the client sends a service request and expects that the results contain only relevant services. Any service that is in the directory response but that does not correspond to the criteria the client specified is useless for the client and ineffectively consumes network resources. The straightforward solution is the worst case for this property since it disregards client criteria and adds all descriptions to the results.

We also analyze the protocol resistance to profiling attacks. This is the privacy property that is the focus of EVEY Light. We want to estimate how difficult it is for an attacker to determine the services that the user is searching from the history of service requests. The straightforward protocol provides the best case for this property: as all announcements are sent back to the client regardless of the service request, the request does not provide any information about the services the client is looking for.

A false positive may happen if two different service descriptions S_1 and S_2 generate the same privacy-enhanced description S_{pe}. Service directories have access only to privacy-enhanced descriptions and thus are unable to differentiate between S_1 and S_2 from S_{pe}. Suppose that the protocol uses SHA-1 [EJ01] as the hash function that generates hashes of inputs, outputs and properties in the case of a syntactic description or that generates the hashes of ontology names for a semantic description. SHA-1 maps entries with arbitrary size to 160-bit strings, so collisions will eventually occur. Even though recent attacks have lowered the bound on the number of messages needed to find a collision in SHA-1 [Coc07], the typical birthday attack for SHA-1 requires generating around 2^{80} messages to find two that hash to the same value [Sch96], which means that for sets containing less than 2^{80} hashed messages, the probability of finding a collision is very low. Moreover, since a service description contains a number of different elements that are independently hashed, a collision between two different descriptions may require the collision of different hashes which reduces even more the probability of false positives. As a result, all service discovery results transmitted are relevant and there is no waste of network resources.

For the discussion on the protection against profiling attacks, we assume that attackers have access to a history of privacy-enhanced discovery requests and privacy-enhanced discovery results, and that they want to identify the service represented by each entry on the discovery history. In particular, the service directory is able to perform this attack since it can easily keep track of the privacy-enhanced service discovery requests issued by

a client and the privacy enhanced service discovery results it receives. As discovery results contain service announcements, the analysis of profiling attacks using the history of service announcements follows directly from the discussion presented here.

If the privacy-enhanced description is syntactic, in order to determine the original announcement or request that produced that description the attacker has to discover the names of inputs, outputs and properties from their hashes. For instance, he must discover n_i from $h(n_i)$, which means that the attacker must successfully execute a *pre-image attack*, which is computationally infeasible for one-way hash functions such as SHA-1. The best strategy for the attacker in this case would be to perform a *pre-computation attack*. To use this strategy, the attacker must have access to every possible service description and generate privacy-enhanced versions of all of them. After that, the attacker simply compares the hashes found on the history with his dictionary of pre-computed hashes to identify the service.

Note however that a pre-computation attack only works if the attacker has access to all possible original service descriptions. If the client is looking for a service unknown to the attacker, the pre-computed dictionary does not contain the service's privacy-enhanced description and thus the attacker is not able to identify the service. Whenever the attacker does not have access to an original description, the only suitable strategy to identify the original service is to run a brute-force attack that enumerates all possible hashes until it finds the one used in the description, which may require the generation of 2^{160} hashed messages. Since clients and service providers no longer need to exchange original descriptions, service descriptions can remain secret to reduce effectiveness of pre-computation attacks.

In the case of a semantic service announcement, the attacker has also access to the concept codes Γ used to annotate inputs, outputs and properties. However, to uniquely define a concept an attacker needs to know not only its code but also the ontology to which the concept belongs. Since the ontology names are also hashed, the difficulty of the attacker to discover the ontology name ω from its hash $h(\omega)$ is the same as discussed above. Concept codes, however, are not uniformly distributed: some codes appear less frequently than others (we present a more extensive discussion of code distribution is Section 4.3.3). This happens because ontology structures are different and concept codes contain structural information. As a result, the attacker can use the concept code to reduce the domain of ontologies of a brute-force attack to find the hash: given a code, some ontologies that do not contain that code can be eliminated. Nevertheless, in order to use this strategy the attacker must have encoded beforehand every possible ontology. This pre-computation attack may be infeasible depending on the number and the size of existing ontologies. Moreover, the attacker must know beforehand the ontologies that clients and service providers use, which is not necessarily the case in pervasive computing.

Privacy Protection Assessment of EVEY Full

The second protocol aims at protecting users from the two types of privacy attacks we are considering: profiling and chosen service monitoring. In this section we discuss how difficult it is for an attacker to discover the services sought by a user based on a history of

discovery requests and results, and how difficult it is for an attacker to know if a client is looking for a specific service also based on a history of discovery requests and results. As EVEY Full by design may include false positives on discovery results, we also analyze how often false positives can happen. Since discovery results contain service announcements, the analysis of attacks aimed at service announcements to invade the privacy of service providers is similar to the one presented here.

Ideally, we want privacy-enhanced descriptions to not reveal any information about the original descriptions that generate them. In other words, we want the probability that the attacker discovers the original description by randomly guessing without any previous knowledge to be the same as the probability of discovering the original description after having access to a privacy-enhanced discovery request or result. However, at the same time, privacy-enhanced service descriptions must contain some information about original descriptions to provide meaningful matching results. Our goal is then to evaluate how much information about the original service description is disclosed and how much remains secret when the attacker has access to its privacy-enhanced version.

Syntactic service discovery requests contain only weak hashes of inputs, outputs and properties names. The amount of information about the original description disclosed by the privacy-enhanced hash depends on two factors: the size of the hash and the distribution of hash results. Smaller hashes provide better privacy protection because the domain of all possible descriptions are mapped to a smaller domain of hashes and thus each hash can potentially represent a greater number of original descriptions due to the higher number of collisions. However, how the hash function distributes the original values into hashes is also important: suppose that a very popular service name is the only one to generate a given hash. In this case, only by knowing the hash value an attacker is able to identify the original service name, and since the service is popular, this may happen quite frequently. The hash function must evenly distribute values from the original domain into the hash co-domain, and must be large enough to be useful for service matching purposes but not as big as to reveal all data about the description for an attacker.

One way to measure how much information is disclosed by the hash function is using Information Theory [Sha48], which defines the concepts of information content, entropy and mutual information. The information content of an event x is a function of its probability and is given by:

$$h(x) = -\log_2 p(x) \tag{4.1}$$

This corresponds to the notion that if an event is certain to happen, it contains no information, while a rare event contains more information. For instance, if a text contains only the character `'a'` and we randomly choose a character from the text, the event that this randomly chosen character is `'a'` has probability 1.0 and information 0 (since we know beforehand that it will happen). The logarithm function uses base 2 so that the result can be interpreted as bits.

The uncertainty related to a random variable X is given by its entropy. The entropy is the sum of the information content of all the possible outcomes of the variable, weighted by the probability that they will happen:

$$H(X) = -\sum_{x} p(x) \log_2 p(x) \tag{4.2}$$

The entropy is maximized when the probabilities of all outcomes are the same. For example, if a text contains every letter of the alphabet with the same probability, we are very uncertain about which character we will choose at random. However, if the text contains 80% of `'a'`s, we are less uncertain about which character will be selected.

In real life, however, a random variable X may be influenced by another variable Y, and knowledge of Y impacts the uncertainty of X. The conditional entropy of X given Y is how much entropy is left in X after Y is learned and it is given by:

$$H(X|Y) = -\sum_{y} p(y) \sum_{x} p(x|y) \log_2 p(x|y) \tag{4.3}$$

Assume that $p(X)$ is the probability of discovering the original service sought by a client through an informed random guess, $p(Y)$ is the probability of discovering the privacy-enhanced service sought by a client through an informed random guess and $p(X|Y)$ is the probability of discovering the original service sought by a client given the privacy-enhanced description he is looking for. By "informed guess" we mean that the attacker knows the distribution of services according to their popularity, and that he can use this information to perform a guess.

We conducted an experimental evaluation using the QWS data set [AMM07], a database that contains 2507 real Web services and their descriptions. For this experiment, we considered only the service name as its syntactic description. The database includes only unique service access points, but since the same service can be deployed in different locations for redundancy purposes, the database contains duplicate service names. We considered that repeated service names represent popular services and are an approximation of the distribution of client discovery requests. We used the `mod` function, which computes the remainder of an integer division as a weak hash function, and we used power of 2 hash sizes from 4 up to 7, resulting from 16 to 128 different hash values.

Figure 4.6 shows the cumulative distribution of service name occurrences in the QWS data set. The greatest part of service names appear only once in the data set (that is the case for 1263 services) while a small number of services appear more than five times on the data set (precisely, 25 services). The maximum entropy of a set with 2507 services happens if they are all equally probable and is equal to $\log_2 2507 = 11.29$ bits. However, service name redundancy reduces the uncertainty of the set and gives an advantage to the attacker. The actual entropy of service names in the QWS Data Set is approximately 10.12 bits, more than one bit smaller than the maximum theoretical entropy. This means that the attacker has average probability of 2^{-10} of correctly guessing the service description sought by a client with a random guess.

Figure 4.7 shows the distribution of service names into hashes for hash sizes of $2^4, 2^5, 2^6$ and 2^7 bits (respectively graphs (a), (b), (c) and (d)). Each column represents a hash value and each color represents the probability of that hash being generated from a different service description. We can see that in graph (a) each column contains more colors than in

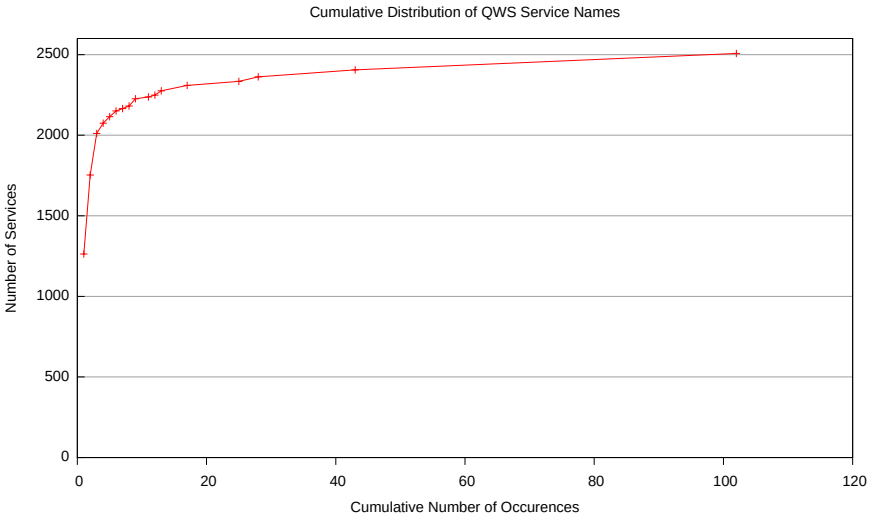


Figure 4.6: Cumulative Distribution of Service Name Occurrences in the QWS Data Set

graph (d), which means that more false positives happen with smaller hash sizes than with bigger. However, since service names in the QWS Data Set are not uniformly distributed, some hash values have a larger probability of representing one service name than others. As we increase the hash size and reduce the number of false positives generated by each hash, we also increase the probability that some hash values univocally identify a service name. As the size of hashes increase, distribution patterns start to emerge until eventually it reproduces the original distribution pattern.

The set of hashes created by a *mod* function using a domain size of 16 bits for the QWS Data Set has approximately 3.97 bits of entropy, while the other sets have respectively 4.95, 5.90 and 6.82 bits of entropy. In this particular case, as the hash distribution and the original distribution are totally correlated, the entropy of the original distribution given the hash is obtained by the formula:

$$H(X|Y) = H(X) - H(Y) \tag{4.4}$$

This means that every bit of information revealed by the hash reduces the uncertainty of the original distribution. In the above cases, the uncertainty that remains from the original descriptions is respectively 6.15, 5.18, 4.22 and 3.30 bits of information. This means that when a 7-bit hash size is used, after learning the hash of a service description an attacker can correctly guess the service it represents with $2^{-3.30} \approx 10\%$ of probability of success.

Service descriptions, however, contain not only the service name, but also its inputs and outputs. Intuitively, as the number of inputs and outputs increase, it becomes more difficult to find a coincidence: it is easier to find two services whose names hash to the same value than to find two service descriptions whose name, inputs and outputs hash to

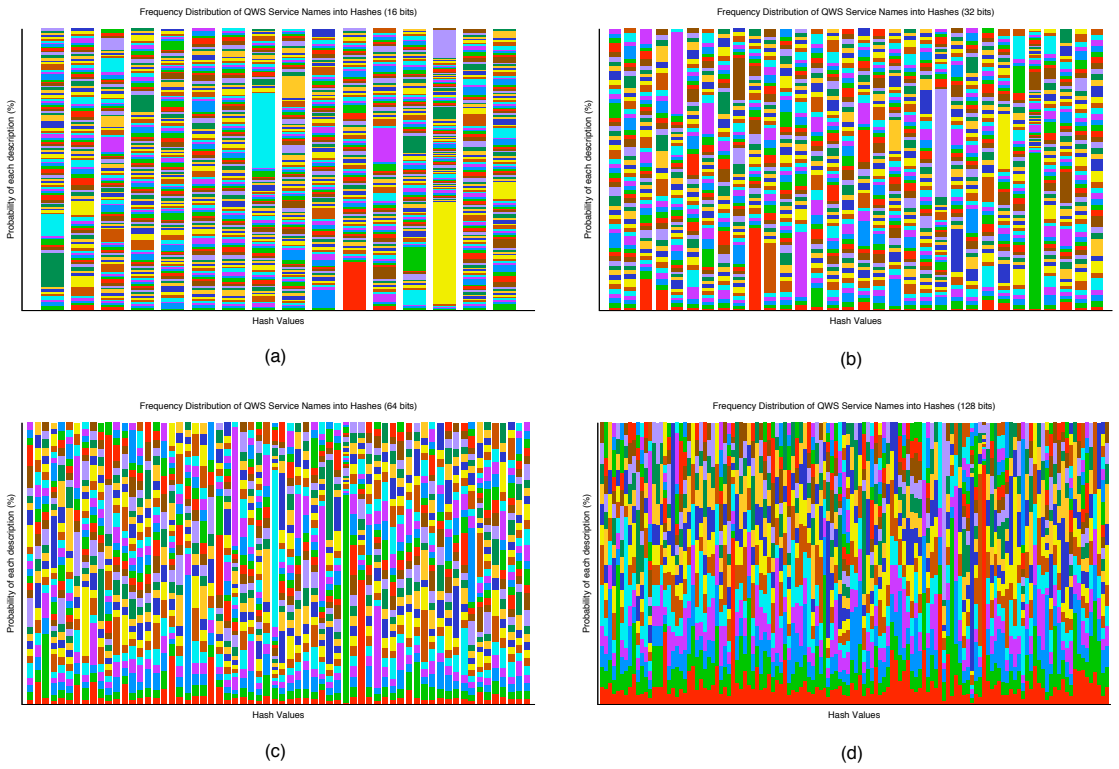


Figure 4.7: Distribution of Service Names in Hashes

the same value. How difficult it is to find coincidences, as the number of service parameters increase depends on many factors such as the average number of parameters in each service description and their distribution when hashed.

Supposing that input names and output names follow the same distribution as service names after hashing, and as we are interested only in the probability of finding exact coincidences, or in other words, we are interested in discovering how easy it is to find two service descriptions that are exactly identical after hashing, we can obtain some approximate results. Table 4.1 shows some probabilities under those assumptions.

When more than one input or output are considered, the probabilities do not necessarily continue to decrease. Suppose that an attacker wants to know which clients are interested in a specific service that requires three inputs and provides three outputs. Discovery requests that match that service include not only descriptions providing the exact three inputs and requiring the exact three outputs, but any other request that requires one or two of the three outputs provided by the service and that provides all three inputs required by the service. A discovery request containing five provided inputs and one required output can match the service that interests the attacker if any three of the five client-provided inputs have the same hashes of the three service-required inputs and the client-required output has the same hash of any of the three service-provided outputs. A similar situation

Event	5-bit Hash	4-bit Hash
Two service names hash to the same value	3.40%	6.55%
Two service descriptions containing the service name and one input hash to the same value	0.12%	0.43%
Two service descriptions containing the service name, one input and one output hash to the same value	0.0039%	0.03%

Table 4.1: Probabilities of Different Coincidences Using 5- and 4-bit Hashes

happens when considering service announcements instead of requests.

From the tests performed with the QWS Data Set, we believe that a hash size of 16 bits provides a reasonable number of false positives and an adequate level of privacy. Additional tests in production environments can help to evaluate if this value is appropriate.

Semantic service descriptions contain the codes of concepts used to semantically annotate the description, without the ontology identification. The protection provided by this technique depends on the number of existing ontologies, the structure of the ontologies and their popularity, or how frequently they are used. Contrary to syntactic service descriptions, semantic descriptions are not yet widely used in real services, so we assume that all ontologies are equally popular.

To evaluate the protection provided by privacy-enhanced semantic service descriptions we encoded 59 existing ontologies using the prime number encoding algorithm proposed by [PB06], and we used two heuristics: root-first which starts encoding the root of the ontology and then follows down to the leaves, and leaves-first which starts by the leaves in the lowest level and continues up to the root.

Figure 4.8 shows the cumulative distribution of codes generated by the set of considered ontologies using both heuristics. The y-axis shows the number of codes that occurred as many times as the number given by the x-axis, or less. For instance, when $x = 10$ we can see that almost 20000 codes happen 10 times or less when ontologies are encoded using the root-first heuristic and around 15000 codes happen 10 times or less when using the leaves-first heuristic. As we can see, when the root-first heuristic is used, the majority of codes occur only once after encoding all ontologies. This is bad for protecting privacy because even though we omit the ontology name in privacy-enhanced descriptions, the concept code can uniquely identify the ontology used since most of the codes are not common to more than one ontology.

Figure 4.9 shows the distribution of unique codes using each of the two encoding heuristics. Using the roots-first, only in 8.24% of the cases the concept code does not uniquely identify the ontology to which it belongs. Most of the time (91.76%) the code is unique for a specific concept on a particular ontology. This means that using the root-first encoding the ontology to which a concept belongs can be identified in more than 90% of the cases by the concept code only. However, when considering the leaves-first encoding

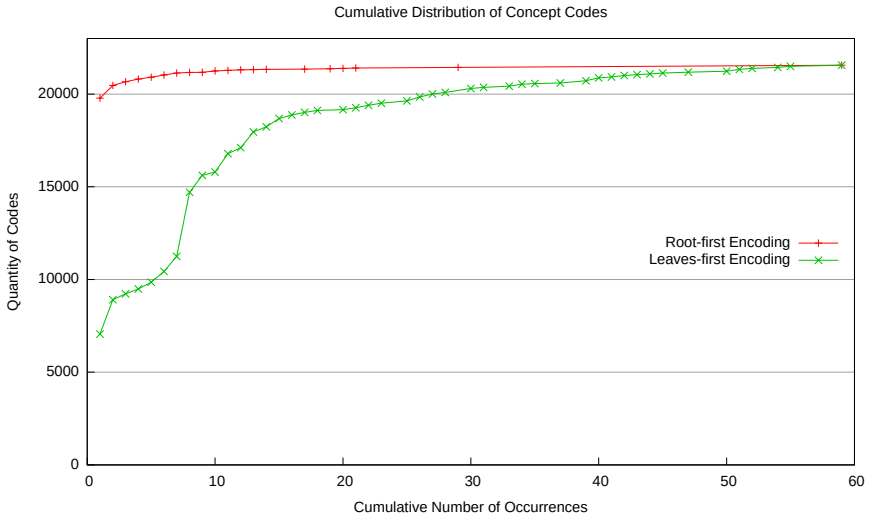


Figure 4.8: Cumulative Distribution of Concept Codes

heuristic only 32.75% of the codes appear on a single ontology, while the greatest part (67.25%) are common to more than one ontology.

The leaves-first encoding strategy causes more coincidences for an additional reason. As a result of how the encoding algorithm uses prime numbers, smaller codes are more common than larger codes. Using the roots-first heuristic, the few coincidences happen with codes up in the ontology hierarchy, or in other words, codes that are more generic. For instance, the code 2 representing the concept `Thing` appeared in all 59 ontologies. Using the leaves-first heuristic, these coincidences happen with codes that represent more specific concepts; codes become less frequent as they are higher on the hierarchy. Even though we do not have real data representing the distribution of ontology codes in actual service descriptions, we can expect that specific concepts are used to describe services more often than generic codes, and thus the number of non-unique codes in real service descriptions using the leaves-first encoding heuristic will be even greater.

If we consider a semantic-annotated service consisting of one input, one output and one category, in approximately 30% of the cases it will contain three non-unique codes. However, the probability of having at least one non-unique code in such a description is of approximately 96%, which is enough to make the description ambiguous. In the case where a service description contains four annotations, the probability that at least one code is non-unique is as high as 98.85% and the probability that at least two codes are non-unique is of approximately 89%.

We must take into account that not all concepts represented by the same code may make sense on a service description, which might give an advantage to the attacker on his task to identify the service that originated a privacy-enhanced description. To measure the number of relevant coincidences, that is the number of concepts that are encoded to the same code and can be exchanged on a service description, we would need a sample

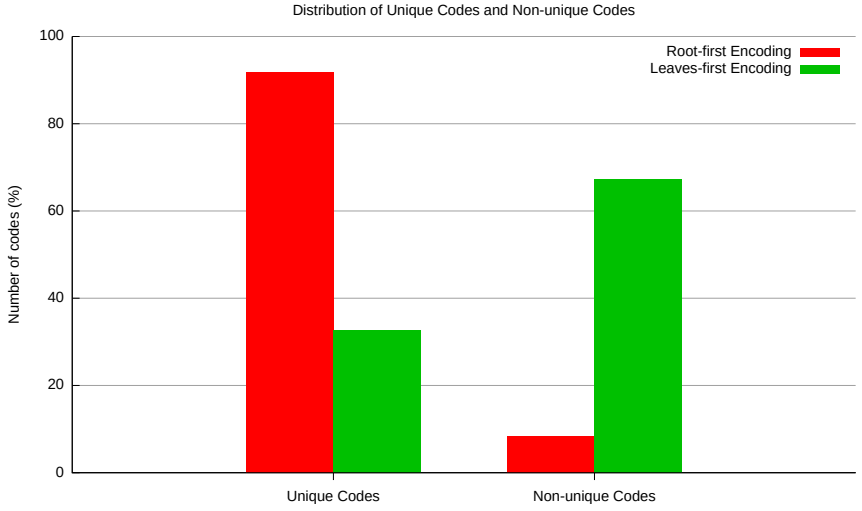


Figure 4.9: Distribution of Unique Codes and Non-Unique Codes

of real semantic annotated service descriptions. Nevertheless, these numbers suggest that the approach is valid and should be tested in real world systems to verify its benefits.

EVEY Performance Evaluation

To validate the efficiency of our solution, we first evaluate the overhead introduced by EVEY, as it is a critical factor for adoption. We more specifically measure the performance costs of creating privacy-enhanced announcements and requests and the performance costs of recovering an original announcement from a privacy-enhanced version. In other words, we evaluate the cost of executing $\mathcal{P}_S$, $\mathcal{P}_S^{-1}$ and $\mathcal{P}_R$ in terms of executing time and file size. We also evaluate the scalability of our protocol comparing the execution time of the privacy-enhanced matching algorithm $\mathcal{M}_{pa}$ with the regular matching algorithm $\mathcal{M}$ for increasing sizes of service directory.

The prototype was developed in Java, using SAWSDL [FL07] to semantically annotate service announcements and requests. Service description files are interpreted using a JAX event-based parser to increase parsing performance. For our tests we assume that cryptographic keys were pre-distributed and agreed upon in advance. We did not consider the overhead of parsing key descriptions nor recovering cryptographic keys. These costs depend on the SAML extension implemented to realize the identity management infrastructure described in Section 4.3.2 and are out of the scope of this thesis.

To evaluate the impact of creating privacy-enhanced announcements and requests, we used service descriptions with a growing number of input and output parameters. We considered service descriptions containing 1 to 16 parameters among inputs and outputs, since it is uncommon to have service descriptions involving more than this number of inputs and/or outputs. All service descriptions were semantic annotated and contained one

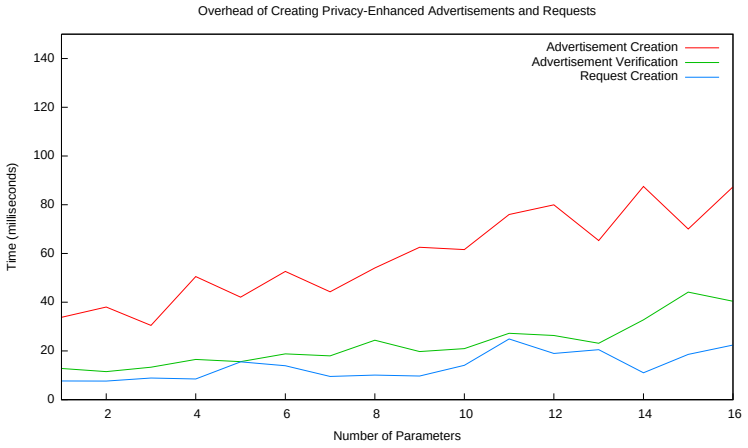


Figure 4.10: Processing Time Overhead

semantic annotation per parameter with each concept belonging to a different ontology. We evaluated the processing time to create privacy-enhanced announcements and requests and to recover an original announcement from a privacy-enhanced version. We also evaluated the file size increase of privacy-enhanced announcements and requests. Tests were performed using symmetric encryption to evaluate the encryption impact on announcements. We used AES with a key size of 128 bits for symmetric encryption. Tests were executed on a PowerBook G4 with a 1.5 GHz processor and 1.25 GB of memory.

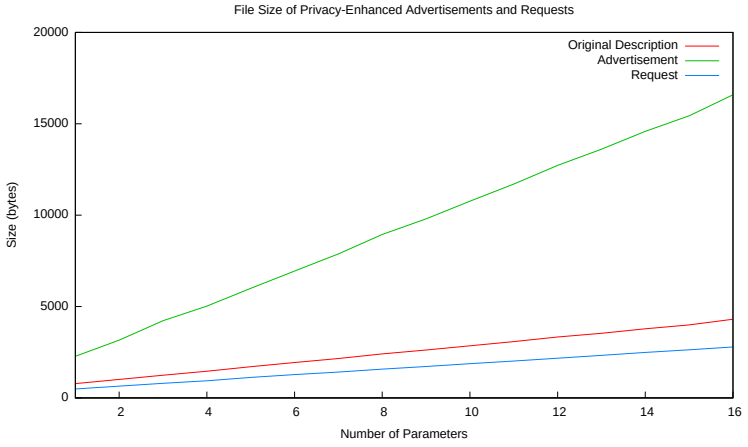
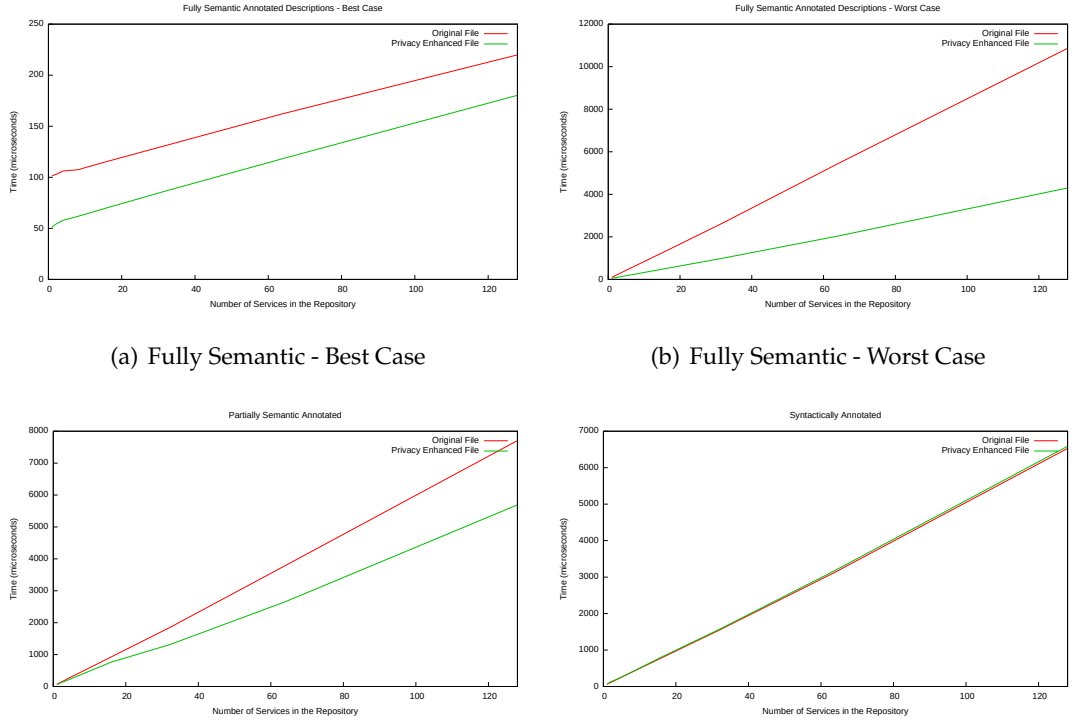


Figure 4.11: File Size Overhead

Figure 4.10 shows the processing time required to create privacy-enhanced announcements and requests and to verify a privacy-enhanced announcement, while Figure 4.11

shows the file size overhead of a privacy-enhanced announcement and request compared to a regular description. The operation that requires more processing time is announcement creation. This is acceptable since service providers do not create announcements frequently and this operation can be performed offline. The other two operations that clients perform at run time, announcement verification and request creation incurred in smaller overheads. To create a privacy-enhanced request for a service containing 16 parameters among inputs, outputs or properties, less than 20 ms is needed. Verification of the same announcement takes around 30 ms, but clients only decrypt a subset of the announcement in discovery results, particularly only the announcements encrypted with keys that the client possesses.

As for the file size, privacy-enhanced requests were actually around 30% smaller than regular requests. This is because we *remove* information from the original request to create a privacy-enhanced request, namely the identification of ontologies used to annotate the description, making it smaller. However, privacy-enhanced service announcements were twice as large as their original counterparts for symmetric and even ten times larger for asymmetric encryption, because our algorithm encrypts all the elements contained on a SAWSDL service description. We can improve it to encrypt only relevant information and thus reduce the size of privacy-enhanced descriptions.



(a) Fully Semantic - Best Case

(b) Fully Semantic - Worst Case

(c) Partially Semantic

(d) Syntactic

Figure 4.12: Performance of Privacy-Aware and Original Matching Algorithms

To evaluate the protocol scalability for growing sizes of data, we analyzed the performance of the privacy-enhanced matching algorithm compared to the performance of the

regular matching algorithm with directories containing from 1 to 128 service announcements. Each test was executed 500 times and average results are reproduced in Figure 4.12. Service descriptions stored in the directory are identical and the request matches all services. This is the worst case for matching syntactic and partially annotated descriptions: if announcements are different from requests, the matching algorithm can abandon the comparison between an announcement and a request before comparing the items on each description if, for instance, the number of inputs required by the announcement is bigger than the number of inputs provided by the request. When all announcements match a request, the syntactic and the hybrid matching algorithm cannot leave the comparison early and must check every element of every description.

For the semantic matching algorithm, however, this is not the case. To improve semantic matching performance, the algorithm proposed by [BKGI06] organizes advertised capabilities that use the same ontologies in graphs such that if the request fails to match the announcement in the root of the graph, it does not need to be compared with the remaining announcements in the same graph. This technique reduces the number of comparisons necessary to check all announcements in the directory. When all announcements are the same, thus, they are all organized by the directory in the same graph and only one announcement is verified, regardless of the number of copies that exist in the directory. As a consequence, for the semantic matching algorithm the worst case happens when all announcements are different. In this case each announcement is classified on a graph of its own and the matching algorithm must check the request against each announcement individually and there is no performance optimization due to the directory organization of announcements. We consider the following cases during evaluation: (a) fully semantic annotated descriptions when all repository descriptions are the same, (b) fully semantic annotated descriptions when all repository descriptions are different, (c) partially semantic annotated descriptions and (d) syntactic service descriptions.

The graphic shows that the performance of the privacy-enhanced matching protocol is similar to the performance of the original algorithm. The time needed to match semantic annotated services in the best case is much smaller than the time required to match partially annotated services or fully syntactic services. For semantic annotated descriptions, the privacy-enhanced matching algorithm performs less comparisons since the ontology names are not part of the request and thus do not need to be taken into account. For syntactic descriptions, the performance is similar since both algorithms must compare a sequence of characters to find a match. The overhead of creating a list containing all matches instead of only finding the match with the minimum semantic distance as it is the case with the original matching algorithm is negligible, because to find a match with minimum distance the original algorithm already computes all possible matches.

To summarize our results, we showed that the biggest overhead in adding privacy enhancements to service discovery is at the service provider side, that have to perform more costly operations and generate larger files. Service publication, however, is uncoupled from service access and this overhead does not directly affect the user experience. Privacy-enhanced service announcements can be created in advance, and since services publication is occasional, the overhead of publishing a larger file is acceptable.

On the client side, privacy-enhanced discovery requests are actually lighter than traditional requests because they contain less information. In EVEY Light, syntactic and semantic descriptions are replaced by their hashes, which are slightly smaller than the original descriptions, while in EVEY Full syntactic descriptions are replaced by a weak hash that is smaller than the original description while semantic descriptions do not contain the ontology identification, which generates files that are 30% smaller in average.

On the other hand, clients must create and verify privacy-enhanced descriptions, which causes a considerable processing time overhead. Although discovery requests need to be created only once and can be reused in later discoveries for the same service, which minimizes the long term impact on performance, clients must verify results after every discovery. This effect on performance can be reduced by using alternative strategies for checking service results such as only checking the results that are potentially closer to the match instead of verifying all results.

Finally, in what concerns service directories, the main overhead is related to storage space and network usage. Since privacy-enhanced announcements are larger than regular announcements, more space is needed to store and more network bandwidth is required to transfer privacy-enhanced announcements. Service directories are generally hosted by devices with more resources and greater network bandwidth than regular nodes, so the architecture can support this additional cost. With regards to processing time, a privacy-enhanced match is faster than a regular match because it involves less comparisons, which reduces the processing load of service directories.

4.4 Distributed EVEY for Pervasive Networks

The EVEY protocol described in the previous section reduces the risk of privacy attacks within a network domain. However, service discovery in HMANETs as defined in Chapter 1, usually comprises multiple networks and involves a hierarchy of service directories. Since every network that is part of an HMANET is independently organized, each network relies on a local directory to store service announcements and handle service discovery inside the domain. At the same time, bridges provide clients with connectivity to distant networks and enables any node to discover and access services in networks nearby to which they are not directly connected.

The problem of enabling service discovery through multiple networks comprising a number of service directories is called **distributed service discovery**. There are basically two approaches to enable distributed service discovery. In the first approach, service directories distribute announcements and synchronize the list of published services. As a result, service directories that are connected to each other contain the same list of service announcements. The second approach distributes client requests to all accessible service directories, and each directory sends results back to the client.

The first strategy unnecessarily creates replication traffic that synchronizes the list of service announcements of each directory, even though replicated service announcements may never match any client request. This additional control traffic is undesirable in

distributed environments containing resource-constrained devices. Moreover, as networks are mobile and dynamically composed, a larger number of messages is necessary to keep directories synchronized as networks join and leave the system. In HMANETs, thus, the strategy of distributing service discovery requests is more effective since it creates additional traffic only when a client is actually looking for a service and announcements on the client's local network do not match the client's requirements.

An alternative approach, as proposed by [SI05], is to combine distribution of service announcements and discovery requests to optimize network usage. In this hybrid approach, directories exchange a summary of the service announcements they contain (in the form of a Bloom filter [Blo70]) and service requests are only distributed to remote service directories that have a high probability of containing a service announcement that matches the client request. This approach requires that service directories perform additional operations before forwarding requests, which may have an impact on their performance. The optimizations introduced by the protocol are particularly appealing in large networks, with thousands of nodes and hundreds of service directories.

In the following sections we assume that distributed service discovery is achieved through distribution of service discovery requests. Whenever a client issues a request, the local service directory matches the request against the service announcements locally stored, and forwards the request to remote service directories if needed. Results from remote directories are sent back to the local directory, that organizes results and sends them back to the client.

Privacy Risks in Distributed Service Discovery

When considering the trust model for HMANETs defined in Chapter 1, distributed service discovery can have a harmful effect on the user privacy even if each service directory implements a privacy-enhanced service discovery protocol such as EVEY. Indeed, clients trust only their local directory to fairly use their personal information since they interact frequently with that directory and are able to measure the directory level of trust through past experiences. Remote directories and bridges, on the other hand, belong to networks dynamically composed that are only temporarily connected to the client. The user generally does not trust these networks and is unable to determine a trust level based on past experiences.

In this setting, broadcast of service discovery requests to all available networks can have an undesirable impact on user privacy. Broadcast of service requests unnecessarily exposes personal information contained in requests to multiple entities of the environment. Even if service descriptions are generalized, as in EVEY, frequent disclosure of discovery requests to potential attackers unnecessarily gives them an advantage when analyzing request data. Exposure of service description data to untrusted entities should be minimized, for instance, by avoiding transmission of discovery requests to untrusted networks if suitable services have already been found.

Access to request data should also be controlled. In pervasive computing, the roles of clients and service providers are not fixed, and users may even perform both at the same

time. This creates yet another possibility for privacy invasions: descriptions in requests and announcements can be combined to infer even more personal information about a user. Untrusted intermediary nodes that are only responsible for routing discovery requests should have access only to information required for routing and must not be able to access request data if this is not necessary.

Finally, untrusted nodes must not be able to correlate consecutive service discovery requests from the same user. Correlation of service requests make it easier for attackers to identify the content of requests and to infer user personal information. In particular, an intermediary node responsible for forwarding service discovery requests can check the source and destination of a message to identify if consecutive service discovery requests are related. It is important, thus, to not disclose the message's source and destination to intermediary nodes responsible for routing to complicate correlation.

4.4.1 EVEY Mechanisms for Privacy in Distributed Service Discovery

To improve privacy protection when distributing service discovery requests to untrusted networks, we propose a set of mechanisms that enable users and service providers to increase privacy protection during service discovery, according to the sensitivity of data contained in the request. These mechanisms aim at reducing the exposure of service discovery requests to untrusted entities thus reducing the access of untrusted entities to the content of service requests. They can be divided into: (i) mechanisms to give clients control over propagation of their service discovery data, presented in Section 4.4.1 and (ii) techniques to increase client privacy during request routing, described in Section 4.4.1.

Control over Discovery Data Propagation

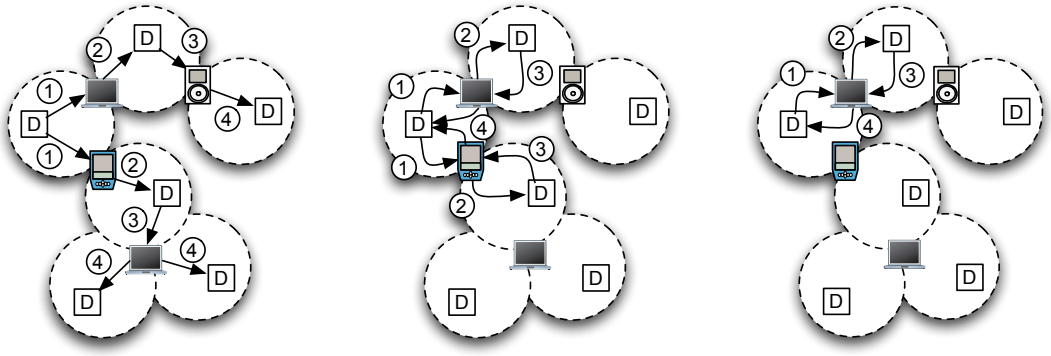
One of the privacy risks in distributed service discovery, described in Section 4.4, is the exposure of service descriptions to untrusted entities in the environment. Those entities can use data from service descriptions to perform attacks using all strategies described in Section 2.1.1. To reduce the risk that clients suffer such privacy attacks, it is necessary to minimize the number of untrusted entities that have access to service request, by controlling the propagation of service discovery messages in an HMANET.

Frequently, privacy protection and performance are conflicting requirements. We believe that the decision to trade-off privacy for performance or vice-versa should not be imposed by the platform but instead should be a user's decision. In this section we introduce optional mechanisms that enable clients and service providers to have more control over service discovery data disclosure, defining *how* requests must be propagated and *where* they should go.

Incremental Service Discovery: We implement in EVEY three strategies to control propagation of service discovery data. The first is the **parallel** strategy, which is simpler and produces faster results, but jeopardizes user privacy. In this strategy, the service

directory in the user local network gradually sends client requests to all connected networks. Discovery requests are quickly propagated to all available networks, but are also exposed to a higher number of untrusted entities. In particular, even if directories in networks closer to the user provide a large number of relevant results for the user request, service discovery request data is still unnecessarily transmitted to all connected networks.

To avoid needless disclosure of service data, EVEY provides two additional strategies that reduces the privacy impact of disclosing service discovery requests to untrusted entities: **progressive** and **one-by-one**. In the former, the local service directory initially sends the request to all the networks that are one hop away and waits for the results. If more results are necessary, the local directory sends the request again to networks two hops away, and so forth. In the latter strategy, the local directory sends the request to a specific directory one hop away and waits for the results. If more results are needed, the request is sent to another directory one hop away and so on, until all directories in neighbor networks are queried. Only then the request is sent to a directory two hops away if more results are still necessary.



(a) Parallel Strategy (b) Progressive Strategy (c) One-by-one Strategy

Figure 4.13: Parallel, Progressive and One-by-one Propagation Strategies

Figure 4.13 shows how the three strategies work. Squares containing the character *D* represent the service directory that stores announcements for the network showed as a dotted circle. Numbers represent the round where the message depicted by an arrow is transmitted. The parallel strategy is asynchronous, so requests are forwarded independently of replies and quickly reach various entities. The progressive strategy waits for results from networks one hop away before deciding if the discovery process should go on. Finally, the one-by-one strategy may disclose the request contents to a single directory, if this directory provides sufficient results.

Clients can define which strategy to use on a per-request basis. When using the progressive strategy, users can inform the number of results that must be obtained before stopping discovery and the maximum number of hops that a discovery request can traverse to reach a remote directory. With the one-by-one strategy, users can also define the maximum total number of directories that can receive the request.

Privacy-Compliant Dissemination: The second mechanism introduced in EVEY allows clients to control where their requests go. Each network dynamically composed in an HMANET provides different properties concerning trust, security and privacy. Users may establish trust relationships with companies or services, and expect those relations to be respected when using pervasive computing systems. The service discovery protocol must enable users to specify such preferences and enforce compliance during service discovery.

Privacy-compliant dissemination of service discovery requests in EVEY is implemented on top of a context-aware mechanism for service discovery, as the one described in [RRdLC+06]. In context-aware service discovery, clients are able to define constraints based on contextual information that controls dissemination of service requests. Nodes responsible for routing service requests first check if the destination complies with the context rules in the request, and only forward the message if the context reported by the destination matches the client-defined rules.

The privacy-compliant dissemination mechanism implemented in EVEY extends context-aware service discovery to take into account two types of trust relationships concerning private data usage: between citizens and governments based on **privacy protection laws** and between consumers and corporations based on **privacy agreements**.

Privacy protection law coverage is related to geographical boundaries, such as state, country, or group of countries. Privacy agreements on the other hand cover relations between a consumer and a company or another user. It is important to notice that, as service requests may traverse several components before reaching the destination service directory, the trust relationship must hold for all the entities on the path, and not only between the client and the destination service directory.

To enable clients to define geographic and domain restrictions for service request propagation in a context-aware service discovery protocol, contextual information must be collected and stored by nodes responsible for handling discovery requests. In the type of environment considered by this thesis, service directories and bridges handle service requests, and so they must keep information about where they are located and the entity (company or user) legally accountable for their operation.

Privacy-related context data is disseminated to neighbor bridges and directories so that they can verify client-defined context rules to decide if the request can be forwarded or not. We developed an elementary ontology for geographical context data specification that increases the flexibility of this mechanism, removing the need for exact matches between context rules and context data (recognizing that a request from a client who trusts the European Union privacy legislation can be forwarded to a service directory in France, for instance).

Service providers can also use privacy-related context data to control access to service announcements. Service providers may want that only clients on a given geographical region under a certain legislation, or that are able to access the network through a path of trusted nodes, are able to discover a service. Service providers thus are able to define privacy preferences in service announcements that prevents service descriptions to reach undesirable clients. Instead of requiring contextual information from clients, and checking during service matching if the client context complies with the provider-defined rules –

which would ultimately reveal the client's location, for instance – contextual preferences from providers are added to discovery results as contextual dissemination rules. As results are forwarded back to the client, privacy preferences are checked and the response may be filtered out at one of the bridges or directories along the path.

Privacy-Enhanced Routing of Service Discovery Requests

We state in Section 4.4 that untrusted components of EVEY responsible for forwarding service discovery requests must not be able to access service data in requests since they may contain personal data. Such components should only be able to access data required for performing routing of service requests. Even though the privacy-enhanced protocol presented in Section 4.3 generalizes service descriptions and reduces their sensitivity, storage and analysis of privacy-enhanced descriptions by intermediary nodes should be avoided to reduce the probability of privacy attacks.

It is also important to protect information concerning message source and destination to complicate correlation of consecutive service discovery requests by untrusted bridges and directories. Our solution for protecting service discovery messages from intermediary nodes is to reveal to each hop only information about the path that is necessary for the message to reach the next hop, and to use encryption between the client's local directory and the destination directory of a service discovery request to avoid that intermediary hops access message data.

The straightforward solution of using public key encryption to protect messages in both directions, however, is unsuitable since it would require disclosure of the local directory identity and reveal the source of the message. We integrate into EVEY an anonymous secure message transmission protocol inspired by the Onion Routing protocol [GRS99] as implemented by Tor [DMS04]. The protocol provides unilateral authentication (request sources remain anonymous), forward secrecy and key freshness. The local directory increasingly establishes symmetric keys with every node in the path to the destination directory, one at a time. Each symmetric key is later used to encrypt the message with different layers of encryption, such that each intermediary node is able to access information from a single layer. Each layer contains only routing information necessary to send the message to the next hop of the path.

If the destination directory possesses a public key and the local directory is able to certify its authenticity, the local directory uses the Tor protocol for establishing a symmetric key. For instance, if directory D_1 wants to establish a key with bridge B_{12} that possesses public and private keys $\{K_{B_{12}}, K_{B_{12}}^{-1}\}$, it first sends a request containing the first half of a Diffie-Hellman key agreement protocol [Sch96] and a circuit ID, encrypted with the Bridge's public key ($D_1 \rightarrow B_{12} : \{g^{x_1}, ID_1\}_{K_{B_{12}}}$). The bridge answers with the second part of the Diffie-Hellman key agreement in plain text, along with a hash of their common key k_1 ($B_{12} \rightarrow D_1 : g^{y_1}, H(k_1)$). The destination directory is authenticated in this case through the public key certificate. If the destination directory does not possess a public key or if the local directory is unable to certify its authenticity, then the local directory can use the protocol for privacy-enhanced service access presented in Chapter 3 to establish a

symmetric key with the destination directory. The network topology in this case indirectly authenticates the destination.

This protocol provides strong anonymity since only the destination directory is capable of accessing the request contents, and intermediary nodes cannot relate the source and destination of a service request. If we take into account the environment dynamics, however, this protocol may be too costly for a single service discovery transaction. Alternatively, local service directories can establish a key only with the final destination directory and use the agreed key to encrypt the service request. In this protocol, however, the relation between source and destination directories of a request is no longer protected. The first node of the path, particularly, knows exactly the source and destination of a service request. Groups of compromised components can also reveal that information. Nevertheless, intermediary nodes are still unable to read the content of service discovery messages.

4.4.2 Distributed EVEY Assessment

Privacy-protection usually has an impact over resource consumption. Some mechanisms proposed in previous sections to improve privacy protection in distributed service discovery, however, allow for a more rational use of network resources and may actually have a positive impact on the performance of EVEY. In this section, we conduct a qualitative and a quantitative evaluation of the privacy-enhancement mechanisms for distributed service discovery proposed here. Whenever it is relevant, we discuss the impacts on users as well as on other components of the service discovery architecture such as bridges and directories.

Qualitative Assessment

The privacy protection mechanisms for distributed service discovery introduced in the last section are not mandatory. Clients can use them independently or in combination to increase control of personal information disclosure in service discovery. As a result, clients that must trade-off privacy for performance are still able to avoid the overhead imposed by those mechanisms.

The parallel flooding strategy is the one that uses more resources from the environment. It generates the largest number of service discovery requests and requires that all available service directories process the request. The other two alternative strategies, progressive and one-by-one, contribute to a better overall usage of node resources by reducing message transmission and processing during service discovery. Experimental data shows that internetwork connectivity degrades substantially after three service directory hops [RAI06]. These two strategies take this result into account and provide a more rational use of resources by performing service discovery first in closer networks, and only forwarding service requests to remote networks if more results are necessary. Even though those strategies take longer to produce discovery results when compared to the parallel strategy, this delay may be acceptable in service discovery since it is a non-interactive process.

Contextual-aware service discovery propagation also helps to reduce the load on bridges

and directories. By checking contextual data before forwarding requests, paths that do not respect client-defined constraints are discarded during request transmission and are not processed by service directories. Concerning service provider privacy preferences, compliance with provider-defined dissemination rules is only checked when results are send back to clients. As a result, some requests may be unnecessarily processed and transmitted. To eliminate this overhead, however, clients would have to disclose some private information to destination directories, which is undesirable. We expect that service providers will define privacy-compliance rules only occasionally and hence this small cost is acceptable. It may be necessary, though, to implement an intrusion detection mechanism to identify bridges and directories that incorrectly report data about location or accountable entities.

Finally, protection of service request content through anonymous encryption can have a significant impact on the performance and response time of service discovery, but it is also optional. Moreover, clients can choose between encrypting messages with the keys from each hop, which causes a larger impact on performance, or only with the key from the destination service directory, which consumes less resources from the environment.

Quantitative Assessment

Each of the optional features for privacy protection in distributed service discovery described in Section 4.4.1 causes an impact on service discovery performance. Depending on the feature, this impact can be experienced only by the client or shared with bridges and service directories. In this section we analyze and detail how those features can influence resource usage in service discovery.

User-defined restrictions on service request propagation based on geographic and accountable entity data are implemented as context rules. Previous results show that service discovery time increases by 1.0 ms to 1.6 ms for each context rule [RRdLC+06]. Based on this data, and since request propagation restrictions can be implemented by two context-rules, we expect that processing of service requests that define restrictions on propagation will be at most 3.2 ms slower, representing a processing time increase of 6.8% on each component per message, which we believe to be acceptable. As service providers may also specify privacy-related context rules, the total delay for processing a service request and its corresponding result is 6.4 ms for each hop and can be at most 51.2 ms for messages passing through 8 hops considering bridges and service directories.

Request propagation strategies can introduce a sensible delay for service discovery. The progressive strategy for service discovery can be much slower than the parallel strategy, especially for networks many hops away. However, for networks one hop away, their performance is identical, since all networks one hop away are discovered in parallel. The one-by-one strategy always performs poorer than the other two strategies, and the delay increases as the number of networks visited by the request also increases. Regarding EVEY components, the progressive and one-by-one strategies may require bridges and service directories to process the same message more than once, when discovering services in networks more than one hop away, but the protocol can be implemented so as to avoid this situation. Nevertheless, the user can choose among the three options which one

Table 4.2: Privacy Performance Overhead

	Client	EVEY	Distributed EVEY
Hop-by-hop Routing (storage)	-	0 KB	3.2 KB
Privacy Related Context	6.4 ms - 51.2 ms (per request)	2.1% - 3.4% (per rule)	6.8% (plus rules)
Progressive Service Discovery	0 - 4 times slower	-	-
One-by-one Service Discovery	0 - n times slower (for n networks)	-	-
End-to-end encryption (weaker)	100 ms (per request)	0 ms (no encryption)	50 ms (Mgrs.) 0 ms (Bridges)
End-to-end encryption (stronger)	200 ms - 800 ms (per request)	0 ms (no encryption)	50 ms

provides the best balance between privacy protection and performance according to the privacy requirements of the discovery request.

Finally, end-to-end encrypted requests using the weaker anonymity protocol require local service directories to perform one public-key encryption and remote service directories to perform a public-key decryption operation per request in the worst case (as symmetric encryption has a smaller cost than asymmetric encryption). Public-key encryption using the RSA algorithm and 1024-bit keys can be achieved in less than 50 ms even in computers with processing power equivalent to today's mobile devices [Wie98]. The processing cost of encrypting and decrypting service discovery requests with the agreed symmetric key is negligible. If the stronger anonymity protocol is used, two public-key operations are required for each message hop in the worst case. As we expect service requests to traverse at most 8 hops (including bridges and service directories), the total encryption overhead for a service request can be as high as 800 ms in the worst case. However, clients can limit end-to-end encrypted service discovery to neighbor networks or networks two hops away, and in that case the encryption cost would be of 400 ms at most per request. Table 4.2 summarizes the costs involved in adding privacy-protection mechanisms for distributed service discovery in EVEY. The table enumerates expected delays for the client and compares the overhead between the native and distributed versions of EVEY.

The proposed mechanisms for increasing privacy protection in distributed service discovery presented in Section 4.4.1 conveniently complement the protocols for privacy-enhanced service discovery presented in Section 4.3. Indeed, distributed service discovery creates additional privacy issues that are not attacked by service discovery protocols that consider a single domain. The extended Distributed EVEY, thus, provide the necessary mechanisms to face those issues and reinforce the privacy protection provided by EVEY.

When considering pervasive computing environments, Distributed EVEY is especially attractive. Pervasive computing integrates multiple administrative domains with mobile hosts. To improve service discovery availability, service directories must be redundantly deployed to cope with the environment dynamics. At the same time, the opportunistic composition of networks complicates creation of trusted environments. EVEY reduces sensible information stored in service directories and thus reduce their trust requirements, while Distributed EVEY protects service discovery traffic among untrusted domains. As a result, service directories can be redundantly instantiated in any available device providing sufficient resources, and users can send service discovery requests to networks dynamically composed without being victims of privacy attacks from malicious nodes.

We got more license plates, wedding gifts, tax returns
Checks to politicians from real estate firms
Money, bills and cancelled checks
We cut relationships with your friends
We're gonna steal your mail!

Dead Kennedys, "Stealing People's Mail"

5

Privacy-aware Service Composition

Service composition is an important feature of Service-Oriented Architectures. It enables users and application developers to use elementary services as building blocks to dynamically create complex composite applications. It also enables clients to elegantly adapt to the environment's dynamics by replacing a faulty or unavailable service with a composition of existing services that provide the same functionality. Service composition leverages the uncoupling of services in an SOA and helps to create applications that are more easily adaptable and scalable.

In the specific case of service-oriented pervasive computing, resources and applications available in a given environment are modeled as services that users may combine to obtain new functionalities. There lies the greatest potential of service-oriented pervasive computing: clients can create innovative and unexpected applications by simply combining available services. Those composite services can be later reused to create yet more novel services, effectively enabling the user to leverage the existing pervasive computing power. Service composition is a key middleware functionality to realize this scenario. Composition mechanisms must overcome challenges such as service heterogeneity and user mobility to enable effortless creation of services that are reliable, secure and that respect user-defined quality constraints.

An often neglected aspect of service composition is its impact over user privacy. In

today's information society, personal information has become a valuable asset and many companies exist whose single purpose is to collect, combine and analyze personal data, threatening civil liberties and the citizen's right to privacy [O'H05]. Those companies explore a known characteristic of personal information: even though a specific piece of data reveals some characteristics of an individual, the combination of multiple sources of personal data provides much more information about the individual than each data analyzed separately.

Information about an individual such as his name, the list of medications he uses, his address and his monthly income reveals more information about the individual when analyzed together than separately. This correlation can have undesirable impacts on the individual's personal life: a loan may be refused for medical reasons or his house may be robbed. There are also situations where personal data must be disclosed together since one helps to contextualize the other. A doctor buying certain drugs may want to disclose also his medical license to avoid misinterpretations.

Service composition is closely related to data correlation. Indeed, when a user creates a service composition, he defines not only the services and the order that they are executed but also the data that is disclosed to each service. As a result, two different service compositions that provide the same functionality can have different impacts on the user privacy depending on which data is disclosed to service providers. A service composition can complicate or facilitate data correlation and ensure that data are disclosed together as well.

In this chapter we discuss privacy issues related to service composition in pervasive computing, and propose a model that allows users to reflect on the privacy impact of executing service compositions. The client can later use this model to compare the effects of executing compositions that are different but functionally equivalent and choose the one that causes less harm to his privacy. We start by discussing the existing models for service composition and the one we adopt for this discussion in Section 5.1. Based on that composition model, we discuss the potential risks to privacy created by composite services in pervasive computing in Section 5.2. Our model to mitigate those risks is based on an extension that we propose to Fuzzy Cognitive Maps (FCMs); both FCMs and our extension – called Fuzzy Cognitive Maps with Causality Feedback (FCM-CFs) – are discussed in greater detail in Section 5.3. After presenting the tools used in our approach, we explain how users can define constraints for personal data disclosure using FCM-CFs in Section 5.4 and how they can use the model to compare the privacy effects of executing a service composition that requires personal data in Section 5.5. Finally, we present a validation of our approach in Section 5.6 and discuss the limitations and possible extensions to the model in Section 5.7.

5.1 Composition Paradigms

The area of software composition is very fruitful and a number of approaches for service composition specification exist in the literature. In the specific case of Web service

composition, some works such as π-calculus [Mil93, LM07] or Petri nets [HB03] enable a more formal definition of service compositions that can be used to verify properties like correctness and liveness. Others are technology-oriented and allow composition based on Web standards, e.g., BPEL [OAS07]. Milanovic and Malek [MM04] provide a comparison of existing solutions for Web service composition in terms of composition requirements like scalability, composition automation and correctness. All those approaches share a process-oriented view of service composition, where the control flow is the primary focus of the composition on top of which the data flow is defined.

More recently, Web mashups [YBCD08] became another popular paradigm for service composition. Instead of focusing in the process part of a composition, mashups propose a data-oriented view of Web service composition. Frameworks for mashup development such as Yahoo! Pipes[1], Google Mashup Editor[2] and Microsoft Popfly[3] enable the user to create new applications that combine data generated by existing services available online. Those frameworks understand the semantics of data used and generated by some web applications and can automatically create the code necessary to feed a service with the output of another. Even though their simple features enable inexperienced users to quickly create new applications through a graphical interface, they lack support for complex control flows. Also, they cannot handle dynamics since mashups are based on a pre-defined list of existing Web services and do not feature mechanisms for service replacement or service discovery in case of failure or unavailability.

When service composition must be performed in dynamic environments where the set of available services change frequently, e.g., in pervasive computing, a new problem emerges: after defining a composition using one of the approaches presented previously, how can we enable the user to realize the composition using only services that are immediately available in the environment? Clearly, if the specification of a service composition is tightly coupled to particular service instances, whenever those instances are not available the user is unable to execute the composition. The middleware must provide a mechanism to users that allows them to specify service compositions that can later be executed with the services readily accessible where the user is located.

There are mainly two approaches to solve this problem: goal-driven service composition and conversation-driven service composition [Ben07]. In the goal-driven approach, the user specifies the characteristics of the task he wants to perform (such as “a payment service on the internet that accepts a certain credit card brand”) and the middleware is responsible for finding one available service or a set of available services that together can perform the user desired task. This specification can be made through a process language (such as PQL in [BK02]) or through a workflow specification language (such as YAWL in [BP05]). The middleware combines available services in every possible way so as all user-provided inputs are consumed and all user-required outputs are generated, regardless of the behavior of the resulting composite service or the conversation that the user defines.

Conversation-driven service composition, on the other hand, allows users to specify

[1] http://pipes.yahoo.com

[2] http://editor.googlemashups.com

[3] http://www.popfly.com

a complex conversation that is used as the basis for the middleware to discover services available in the environment. Sirin *et al.* [SHP03] propose an approach where the user manually composes a service based on the characteristics of available services. Composition happens in phases: the user selects one of the available services and on the next phase the middleware shows to the user only services that are relevant to the current composition. Ben Mokhtar *et al.* [BGI06] propose an automated approach to conversation-driven service composition that attempts to realize the exact conversation that the user specifies, and if this is not possible the middleware automatically recomposes the conversations of available services to propose alternative executable workflows that realize the user-defined task.

In what concerns the execution of composite Web Services, two approaches are considered [SH05]. **Orchestration** assumes that a single entity is responsible for calling each service of the composition, sending inputs to services and receiving their output, similar to a maestro conducting an orchestra. **Choreography**, on the other hand, assumes that services are autonomously organized to perform the client-defined composition, passing control and data to each other as needed, in a way similar to how dancers organize themselves on a stage. While orchestration may create an overhead on the orchestrator, choreography is harder to implement since it requires that services are aware of each other.

We believe that orchestration is an execution paradigm more adapted to the openness of pervasive computing scenarios, where the set of available services on an environment changes frequently and services are not necessarily aware of the existence of other services. We assume that the middleware installed on each device of the pervasive computing environment provides a mechanism for conversation-driven service composition such as the one in [BGI06]. Based on this feature, clients describe service compositions by defining **abstract workflows**, possibly through a graphical interface. An abstract workflow describes the process to obtain a certain functionality including its control and data flows. Abstract workflows can be built based on the user's past experiences creating composite services or using directions provided by other users or service providers.

Unlike traditional workflows, an abstract workflow can be instantiated by different **executable workflows**, which are instances of the abstract workflow that can be realized using available services. These workflows can be created based on the actual services available on the environment. The middleware's service composition component relies on the service discovery component to identify available and appropriate services. The service composition component is also responsible for re-defining the user-provided workflow into another workflow that provides the same functionality required by the user but is adapted to the characteristics of the user current environment. The service composition mechanism may divide one service of the abstract workflow into two or more services in the executable workflow, or it may merge different services from the abstract workflow into a single service in the executable workflow.

Throughout this chapter we will use the YAWL notation [VT05] to represent the control flow of a workflow, and incoming and outgoing dashed arrows to represent the data flow. We use YAWL because it is a formal language for workflow description that supports a number of workflow patterns, it is visually easy to understand, flexible, and becoming widespread. Moreover, there is ongoing work into transforming BPEL processes

into YAWL notation, which would encourage the integration of this language into the Web Services architecture [BP06].

5.2 Privacy Risks in Service Composition

The adoption of a service composition paradigm as the one described in the previous section raises some privacy concerns. Definition of a service composition workflow involves the definition of a control flow, that specifies which services can be executed in parallel, what to do if the execution of a service fails, how to select a service among some options, synchronization between flows of execution, among others. Workflow patterns [AHKB03] extensively list the constructions that can be part of a control flow. Users must also define the data flow, which specifies how data is transmitted among services and which data is accessed by each service. Even if users do not explicitly define a data flow, the composition control flow induces the data flow used by the composition.

Definition of how data is partitioned among service providers is fundamental for privacy protection. This problem was initially identified by the Chinese Wall Security Policy [BN89], that has been used for many years to ensure that entities do not access data with conflicting interests. The Chinese Wall is a dynamic access control policy that defines possible accesses in the future based on data already accessed in the past. This policy was created to avoid that financial analysts have access to data from competing companies since correlation of this data creates a conflict of interests.

Data correlation is a serious privacy issue in our present society. Different private organizations such as banks, credit companies and internet service providers, as well as governmental institutions such as revenue services, healthcare system and transportation authorities keep electronic databases containing consumer and citizen personal data. Correlation of those multiple databases may reveal information that individuals never wanted to disclose, without their awareness. For instance, even though one's gender, zip code and date of birth may not reveal much information individually, researchers affirm that between 63% [Gol06] and 87% [Swe00] of the American population can be uniquely identified when combining those three attributes.

Sometimes private data is naturally conflicting and can expose crucial personal information when correlated. Imagine that Lucy is at the airport waiting for a flight when she realizes that she forgot to bring her medication. Instead of walking around the airport in search of a pharmacy, she decides to use her mobile device to execute the service composition defined by Figure 5.1. She looks for a pharmacy to place her order, receives an invoice that she pays at her bank, and uses the airport location service to find the pharmacy and pick up her medication.

Lucy may not feel comfortable to share her list of medications with her bank, since knowledge of this data could influence her bank's rates and terms for a personal loan, for instance. This can happen if the invoice from the pharmacy contains the list of medications she bought, or if her bank also provides the pharmacy service. Correlation among different data increases the quality and precision of profiles, and thus its value. This financial

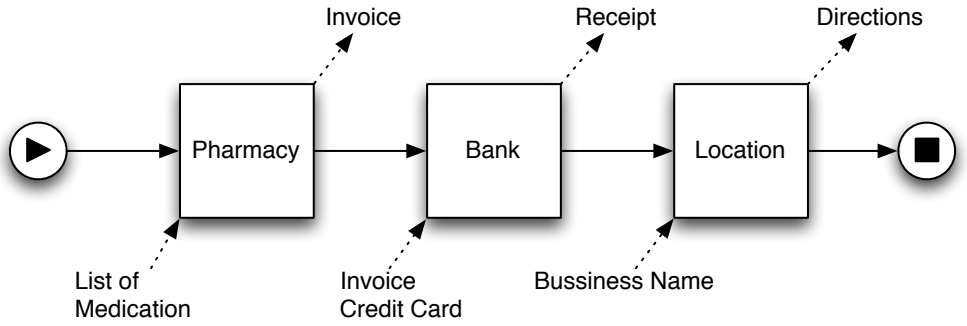


Figure 5.1: A Privacy Sensitive Service Composition

possibility was already noticed by the private sector, and today there are companies selling services that explore correlation of seemingly harmless data to create elaborated user profiles [O'H05].

The potential of obtaining money from personal information creates a financial incentive for theft of databases containing personal data. Indeed, databases containing detailed personal information can help thieves to perform frauds that are more likely to succeed. Not surprisingly, the number of security incidents that result on leakage of individual's personal data keeps increasing (such as [The07, BBC07, Cam08] to name a few). Information contained in such databases is so valuable for creating successful scams that criminals are willing to pay to legally obtain access to personal information databases. There is at least one known case where a data collection company (ChoicePoint) purposely sold to a group of thieves a database containing personal data from 165,000 individuals [Con06].

Pervasive computing environments are a valuable source of personal information since individuals are expected to use pervasive services to perform daily tasks and to generate a great quantity of virtual imprints related to their activity. Such data, if extensively collected and analyzed, can be used to infer a number of intimate traits and beliefs that individuals never meant to disclose publicly. Users, thus, must be aware of the data correlation risks involved in executing a service composition.

There are two main reasons why users are currently unaware of the correlation risks regarding the execution of service compositions in pervasive computing environments. The first reason is that most of the existing solutions for service composition have a process-oriented view of service composition and thus users focus more on the control flow. Data-oriented composition techniques such as Web mashups do not emphasize the correlation potential between data nor the risk of correlation as a consequence of executing a service composition.

The second reason is that service composition in open and dynamic environments requires from the middleware a flexible composition mechanism that can adapt the user-defined workflow to use currently available services. This adaptation is also focused on the composition functionality and may change data partition as explained in the previous

section. As a consequence, even if the user specified a data flow respecting certain privacy requirements, he has no guarantees that the data flow will be respected by the executable workflows proposed by the middleware.

We believe that the specification of allowed data flows in the abstract workflow of a service composition is not the most suitable technique to avoid data correlation in pervasive computing. When considering the data required for executing a service composition, there is potentially an exponential number of possibilities that services can be reorganized to access different sets of disclosed data. To specify the data partitions that respect the user privacy it is necessary to evaluate each possible partition. If not all partitions are specified, it is possible that the middleware proposed workflow contains a valid but not specified data partition and the user will not be able to execute it.

Since pervasive computing environments are dynamic and users do not know beforehand the services that will be available, it is very unlikely that the middleware will find the exact user-defined composition. To increase the probability that the user realizes the intended composition, our approach does not try to limit the number of service compositions that the middleware can find. Instead, we propose a model for reasoning about the privacy impact of executing a service composition that can be used to evaluate the executable workflows proposed by the middleware. Based on the risks associated with each available executable workflow, users can select the one that causes the smallest effect on his privacy, or avoid their execution if they are too risky. Our approach is based on an extension to Fuzzy Cognitive Maps which are detailed in the following sections.

5.3 Using Fuzzy Cognitive Maps to Model Privacy Impact

In this section we introduce Fuzzy Cognitive Maps (FCM), the tool we extend to model the privacy impact of service compositions. Our extension enables users to reason about the effects to privacy of executing different but functionally equivalent service compositions. We present FCMs in Section 5.3.1 and discuss some topics that arise when using FCMs to model real world situations. After that, in Section 5.3.2, we introduce our extension to FCMs that overcome some of the limitations of the original model. We justify the need for this extension in Section 5.3.3, showing how the results obtained using our extension are different than the results of representations that use the original model to obtain the same effect. We finish by discussing other existing extensions to FCMs in Section 5.3.4, emphasizing how they fail short to address the issues that lead us to propose our extension.

5.3.1 Fuzzy Cognitive Maps

Fuzzy Cognitive Maps were proposed by Kosko [Kos86] based on the work developed by Axelrod [Axe76] on cognitive maps to model social knowledge. Cognitive maps are graphs that describe causality (edges) among varying concepts (nodes). A positive edge between two concepts means that increasing the first concept will result in increasing the second and conversely a negative edge indicates that increasing the first concept will cause a

decrease on the second.

Kosko noticed that causality generally happens in degrees, and the traditional model of cognitive maps, where edges can assume only one value in $\{+, -\}$, was too rigid to represent real-world reasoning. To overcome this limitation he introduced fuzziness in cognitive maps, allowing the representation of causality with different levels (such as *a little*, *much* and *a lot*). Formally, an FCM is a fuzzy graph containing N nodes $C_1, ..., C_N$ representing concepts, and N^2 edges $W_{i,j}$ representing causality between concepts C_i and C_j. Concepts C_i are fuzzy sets and can assume values in the interval $[0, 1]$ while edges $W_{i,j}$ can represent positive or negative causality between concepts i and j receiving values in the interval $[-1, 1]$. The initial state of an FCM contains the initial values of each concept, $S(0) = \{C_1(0), ..., C_N(0)\}$. Figure 5.2 shows an FCM with 5 concepts (denoted by circles) and 8 edges (with causality values in boxes).

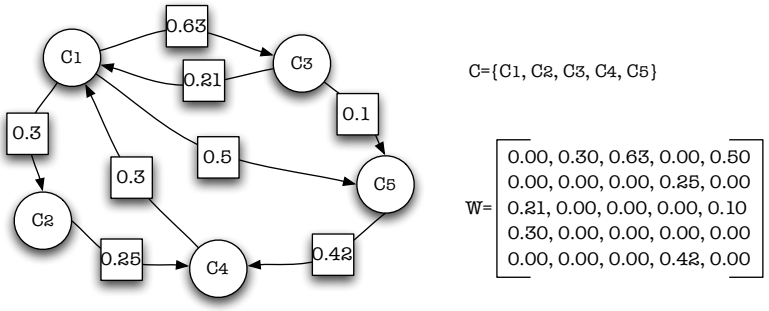


Figure 5.2: A Simple Fuzzy Cognitive Map with its Set of Concepts and its Weight Matrix

Fuzzy Cognitive Maps can be used to answer *what-if* questions related to the concepts represented. Given an initial state $S(0)$ and a matrix W containing the values of all edges $W_{i,j}$ between concepts on the FCM (where $W_{i,j} = 0$ means that there are no edges between concepts i and j), state $S(t)$ can be obtained by the matrix product $S(t-1) \times W$. To better represent learning, however, a sigmoid function is used to reduce the amplitude of the new value. Equation 5.1 is normally used to compute the next state of a concept on an FCM, where f is a sigmoid function such as arctangent. This operation is repeated until the map reaches a stable state (either $S(t) = S(t-1)$ or a cycle of states is detected).

$$C_i(t) = f(\sum_{j=1}^{N} W_{j,i} C_j(t-1)) \tag{5.1}$$

FCMs proved to be a good abstraction to identify hidden patterns in causal relations and to simulate the global effects of changes in the values of some concepts. FCMs have been used to model causal relations in society [KLC04], international politics [Per96], control engineering [SG00] and virtual worlds [DK93]. Researchers developed learning algorithms for FCMs that fine-tune edge weights to lead the FCM to certain pre-defined steady states [CF96]. However, FCMs have some weaknesses such as lack of support for

time, conditional weights and non-linear weights [Hag92].

Which Sigmoid Function to Use?

Most descriptions of a Fuzzy Cognitive Map propose the usage of a sigmoid function [Kos91] or a bounded signal function [DK93] during next step computation to better model knowledge acquisition. This idea is also used in Neural Networks, and its goal is to better simulate non-linear systems such as the human brain.

Two functions are mostly used as a sigmoid function to compute the next state of a concept based on the total stimulus it receives in fuzzy cognitive maps: the logistic function (5.2) and the arctangent function (5.3). Neural networks, on the other hand, propose the usage of the hyperbolic tangent (5.4). Plots for all three functions are given in Figure 5.3.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{5.2}$$

$$f(x) = arctan(x) \tag{5.3}$$

$$f(x) = tanh(x) \tag{5.4}$$

A sigmoid function for FCMs has some requirements. First of all, concept values in FCMs are fuzzy and thus must be in the interval [0,1]. As such, the sigmoid function codomain should be limited to this interval. Moreover, the next step value of a concept should be proportional to its inputs: small inputs should generate small outputs. As we can see from Figure 5.3, neither the logistic nor the arctangent functions are suitable for FCMs requirements, since the arctangent function generates negative values for negative inputs and the logistic function generates values above 0.5 even when the input is close to zero.

We performed tests using different versions of all three functions and the one that provides results closer to expected is the hyperbolic tangent function. However, to keep results consistent with the fuzzy framework, we consider every negative result produced by the hyperbolic tangent function as 0.

How to Model Learning?

Even though FCMs are used in a wide range of applications in different areas, there is a certain disagreement concerning the next step function defined in Equation 5.1. This function represents each node's ability of "learning" from the values of other concepts in the FCM. In every iteration each node suffers an influence from other nodes; the next step function models how this influence impacts a node after each iteration.

We found in the literature three different formulas for computing the next step of an FCM. Equation 5.1 is used in [SG99], and considers the new value of a concept only as

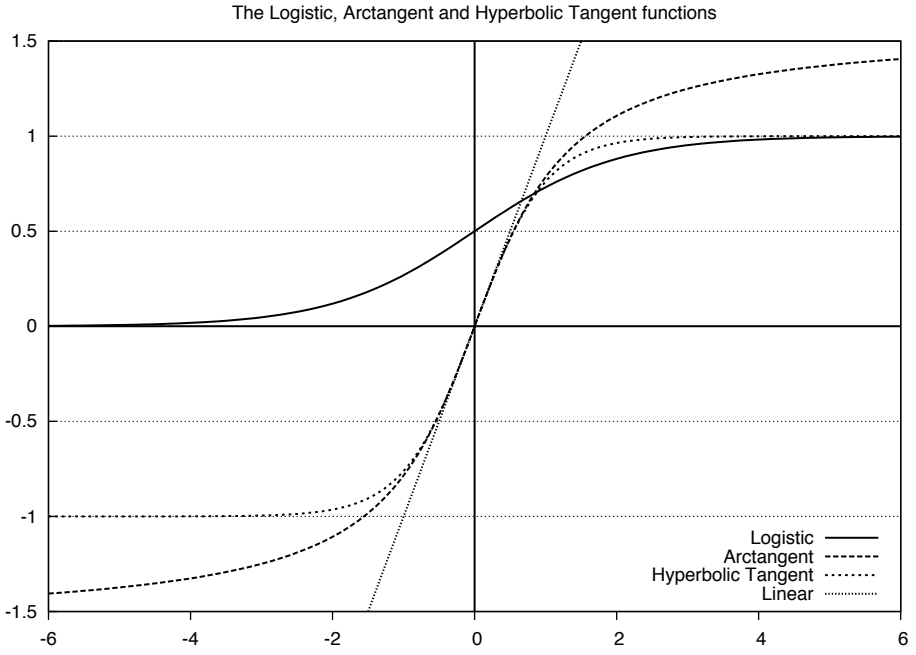


Figure 5.3: The Logistic, Arctangent, Hyperbolic Tangent and Linear Functions

a function of its inputs, without taking into account the previous concept value. Equation 5.5, used in [SGG97], computes the new value of a concept based on its previous value, but applies the sigmoid function only to the offset due to the next step. Finally, Equation 5.6 specified in [PSG06] also takes into account the node's previous value but applies the sigmoid function to the sum of the previous value with the offset of the next step.

$$C_i(t) = C_i(t-1) + f(\sum_{j=1}^{N} W_{j,i} C_j(t-1)) \tag{5.5}$$

$$C_i(t) = f(C_i(t-1) + \sum_{j=1}^{N} W_{j,i} C_j(t-1)) \tag{5.6}$$

To understand the differences between those functions and to evaluate their behavior, we applied each function to a very simple FCM, represented in Figure 5.4. Equation 5.5 can potentially generate values greater than 1 because the sigmoid function ensures that the offset is smaller than 1, but adding the offset to the previous concept value can result in a number bigger than 1. In order to keep the valuer of $C_i(t)$ fuzzy, i. e. in between 0 and 1, we force the new value to be at most 1 in those cases.

When applied to the map of Figure 5.4, Equation 5.5 saturates all the concepts and their final value is 1. It is easy to see that, when using this equation, the values of any

concept suffering a positive causal relation will increase continuously until it reaches 1, unless the causality source is zeroed. This happens because the sigmoid function does not attenuate the previous concept value, and the concept's new value is the result of adding something to the previous value in every iteration.

Equation 5.6, on the other hand, applies the sigmoid function to the total result of adding the previous concept value to the next step increment. The result is that the concept value no longer increases indefinitely but eventually stabilizes in a value different than 1. In the example of Figure 5.4 and using the hyperbolic tangent as the sigmoid function, the values of the three concepts were respectively 0.63, 0.77 and 0.57, which gives more information about how the map reaches an equilibrium than the previous equation.

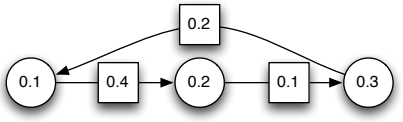


Figure 5.4: A Simple FCM

Finally, Equation 5.1 considers as the new value of a concept only the result of the inputs it is receiving under that map iteration, discarding any previous value. The result for the map of Figure 5.4 is that the values for all concepts will quickly approach zero since each concept suffers a causal relation from a single other concept and this relation attenuates the concept value of the causal source.

Even though in this example Equation 5.6 gave the best results, in general this is not the case. Two characteristics of the map in Figure 5.4 influenced this result. First, all concepts in this map suffer causal effect. When a concept does not suffer causal effect and its next step value is computed using this equation, the concept will eventually reach 0 since $tanh(x) < x$. Second, each concept suffers a small causal effect produced by a single concept. As such, the sum $\sum_{j=1}^{N} W_{j,i}C_j(t-1)$ results in a small number, and adding it to the previous concept value does not saturate it. However, as the number of causal effects suffered by a concept increases, this equation tends to quickly converge to 1 and may produce unrealistic results. This issue has already been mentioned by Carvalho and Tomé [CT02] who state that it is impossible to model in FCMs a situation where $C(t+1) = C(t)$ for every possible initial value $C(0)$, unless a linear function is used.

For those reasons, in our simulations we use Equation 5.1. Whenever we want to consider the previous concept result when computing its new value, we add an edge coming from the concept and directed to the concept itself with value 1.0. This allows us to remove from the FCM situations where this "memory" of previous values is undesired. This equation however still suffers from the same issue as Equation 5.6 when computing the next step value for concepts without causal relation: their values eventually converge to zero. When using FCMs to model the privacy impact of disclosing private information,

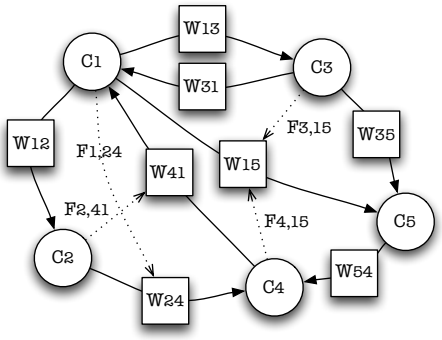


Figure 5.5: An Example of a Fuzzy Cognitive Map with Causality Feedback

this behavior is undesirable. In our model, thus, whenever a concept does not suffer causal effects we consider that its next value is the same as its previous value.

5.3.2 Fuzzy Cognitive Maps with Causality Feedback

Despite their power, FCMs fail to capture an important notion of private information disclosure: the impact of disclosing a personal attribute to an entity increases the privacy impact of disclosing other attributes to the same entity depending on how those attributes relate. Consider the user's login name and his pseudonym (that he uses, for instance, to post messages to a forum). Disclosure of his pseudonym causes a small impact on his privacy, since his identity remains protected. However, if the user has already disclosed his name to an entity, the impact of revealing his pseudonym to the same entity is much larger because it allows correlation of his pseudonym to his identity.

In the example above, the causal relation between disclosure of the pseudonym and privacy impact changes according to other data already disclosed to the same entity. There is a relation among concepts and causality that cannot be modeled by traditional FCMs. To enable the expression of such relations, we propose an extension to FCMs named Fuzzy Cognitive Maps with Causality Feedback, or FCM-CF.

An FCM-CF is the combination of two graphs. The first is a traditional FCM with N concepts $C_1, ..., C_N$ and N^2 edges $W_{i,j}$ representing causality between concepts C_i and C_j, where each C_i can assume values in the interval $[0, 1]$ and each $W_{i,j}$ can receive values in the interval $[-1, 1]$. The second graph represents causality feedback and has two sets of nodes: *source* nodes and *destination* nodes. Source nodes are the concepts C on the FCM and destination nodes are the edges W on the FCM. Edges $F_{i,jk}$ on the causality feedback graph can assume a value in $[-1, 1]$ and always connect a source node C_i to a destination node $W_{j,k}$. As a result, a causality feedback graph CF=$(C \cup W, F)$ can have at most N^3 edges. Figure 5.5 shows a simple FCM-CF graph, where solid lines are edges on FCM while dashed lines are edges on CF.

The state of an FCM-CF at step t is composed of the values of its concepts and its

weights on instant t, $S(t) = (C(t), W(t))$. Next state computation is a two-step process in FCM-CFs. Given a state $S(t)$, the values of $C(t)$ and $W(t)$ are fed into the CF graph to obtain the new matrix $W(t+1)$ that considers the causality increase on the FCM caused by the values of concepts. New values for W are computed using Eq. 5.6 instead of Eq. 5.1 which is used in the FCM. This is because a causality feedback edge models the *increase* in causality produced by a concept. This value must be *added* to the already existing causality and should not replace it, as it happens when using Eq. 5.1. After computing the next step values for the CF graph, the new FCM= $(C(t), W(t+1))$ is used to compute $C(t+1)$. In summary, for each state containing the values of concepts and the weights on an FCM, first the edges of the FCM are updated according to the CF graph, and then the new concept values are obtained through the FCM.

5.3.3 Motivating Fuzzy Cognitive Maps with Causality Feedback

One concern that arises whenever an extension to an existing model is proposed is to know if that extension is absolutely necessary. An extension to a model can be justified if it introduces new features or if it simplifies the representation of a number of situations comparing to the original model. In the case of FCM-CFs, we are interested to know if the same behaviors that can be specified using FCM-CFs can also be defined using elementary FCMs and if so, how easy it is to construct an FCM that works as an FCM-CF.

Our first attempt to mimic the behavior of FCM-CFs using FCMs is to replace the causality feedback edge with a new concept and two edges: one from the source of the causality feedback edge to the new concept and another from the new concept to the destination of the causality feedback edge. In order to compare the evolution of node values in different maps, we simulate three different maps, an FCM, an FCM-CF and an alternative representation of the FCM-CF using an FCM. Figure 5.6 shows the FCM without causality feedback, Fig. 5.7 shows the same map with a causality feedback edge coming out of node 3 and Fig. 5.8 shows an alternative representation of the FCM-CF using a basic FCM.

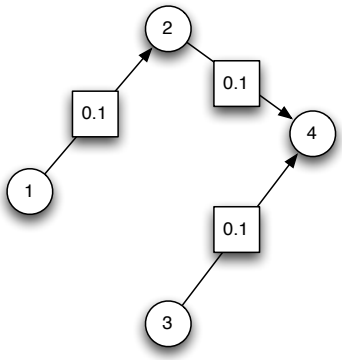


Figure 5.6: FCM

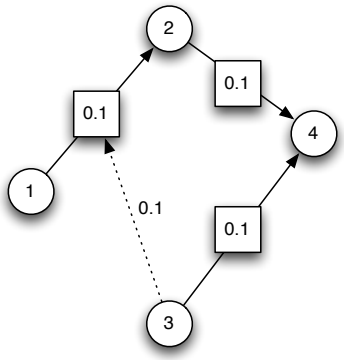


Figure 5.7: FCM-CF

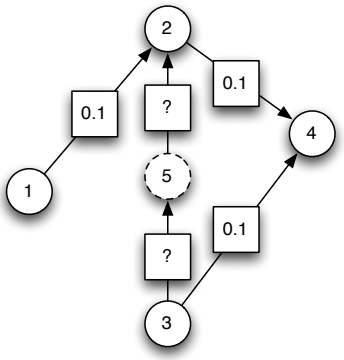


Figure 5.8: Alternative

The first issue when replacing a causality feedback edge by a new node is which value the new edges introduced on the map must have to produce the same effect. Those edges are marked with `'?'` in Fig. 5.8. In our simulations we used three different values for those edges: 0.1, 0.13 and 0.3. The initial value of all nodes is 0.1 and we are interested on the values of node 2, the one suffering the effects of causality feedback. Figure 5.9 shows the evolution of the value of node 2 until it stabilizes and Fig. 5.10 shows the final state of all nodes on the maps (except node 5 that only appears on the alternative representation).

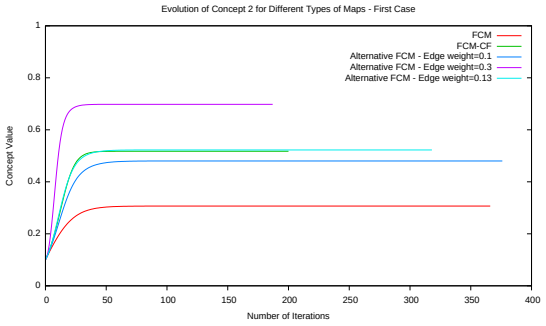


Figure 5.9: Node 2 Evolution - First Map

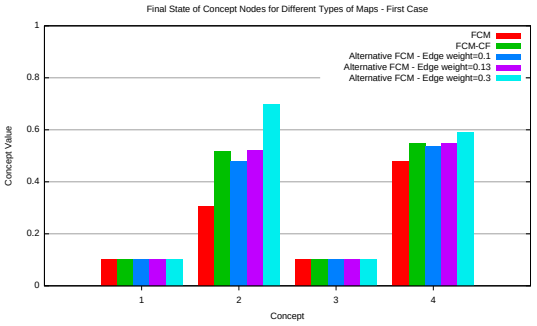


Figure 5.10: Final States - First Map

From those graphs, we can see that the results obtained with the alternative representation using an edge value of 0.13 is reasonably similar to the results generated by the FCM-CF. Even though the alternative representation required more iterations to stabilize, the final states are comparable in both cases. This first map, however, is too simple. The node at the source of the causality feedback (node 3) does not suffer any influence from other nodes and its value is constant throughout all iterations. The following map is a bit more complex and allows us to compare the results with those obtained for the first map. Here again Fig. 5.11 represents an FCM, Fig. 5.12 shows the same FCM with one causality feedback edge and Fig. 5.13 shows the alternative representation of an FCM-CF using only an FCM. We simulate the alternative representation using values 0.1, 0.13 and 0.3 for edges marked `'?'`.

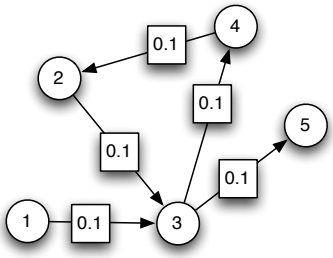


Figure 5.11: FCM

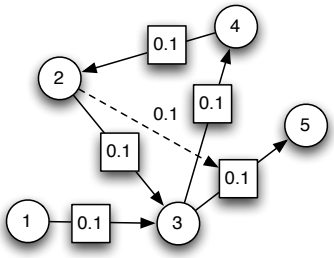


Figure 5.12: FCM-CF

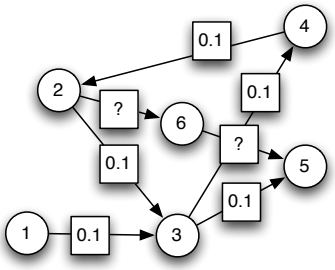


Figure 5.13: Alternative

We are interested in evaluating the evolution of concept 5, the one suffering the effects of causality feedback. Simulation results are presented in Fig. 5.14 and Fig. 5.15. We can see that in this case the alternative representation using edge values of 0.13 did not have the same behavior as the FCM-CF. The value of concept 5 in the FCM-CF was closer to the alternative representation using edge values of 0.3 for the second map. Still, the value evolution for concept 5 was different in both cases: it increased faster in the alternative representation than in the FCM-CF.

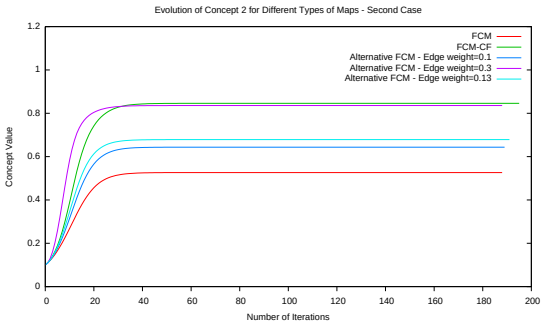


Figure 5.14: Node 5 Evolution - Second Map

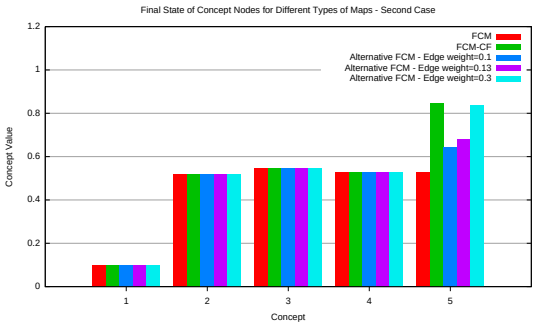


Figure 5.15: Final States - Second Map

This shows that an FCM-CF and this type of alternative representation are not directly equivalent. It may be possible to create "conversion rules" from one representation to another, attributing values to edges according to the characteristics of maps (such as the presence of cycles or if the source of a causality edge is constant), but this rules could become unfeasible depending on the size and complexity of the original FCM-CF.

5.3.4 Other Extensions to Fuzzy Cognitive Maps

Other researchers before us also identified shortcomings when using FCMs and proposed extensions to make this model more flexible. Carvalho and Tomé proposed the use of Role-based Fuzzy Cognitive Maps [CT99] to express conditional causal relations such as "if concept A increases a little than concept B decreases a lot". They showed that some FCMs that diverge can be made to converge when modeled with RBFCMs.

Hagiwara identified three weaknesses in FCMs, namely, lack of support for non-linear causal relations, absence of a mechanism to represent time and the impossibility to model conditional relations, that only exist according to another concept. He proposed Extended Fuzzy Cognitive Maps (E-FCMs) [Hag92] to solve those issues. E-FCMs allow causal relations to assume values defined by a non-linear function, introduces a new concept called *delay* that holds back the effects of a causal relation until computation of a certain map iteration t, and enables modeling of causal relations that are only active if a certain condition is observed.

Miao *et al.* consider that E-FCMs are a naïve extension to FCMs that fail to model system dynamics and suggest an extension called Dynamical Cognitive Networks (DCNs) [MSLL99].

In DCNs, the weight of a causal relation can increase as time passes, which allows modeling of more complex situations that are not possible with E-FCMs which only enables or disables a causal effect after some iterations. Also, DCNs models concept values as a set of possible values instead of a continuum, and as a consequence the value of a concept after considering its causal relations is not the direct result of the next step computation function but instead one of the values in the set according to a membership function.

Finally, Aguilar introduced Adaptive Random Fuzzy Cognitive Maps (ARFCM) to add dynamics to the weights of causal relations [Agu02]. ARFCMs use the ideas of the Random Neural Network Model to strengthen some causal relations and weaken others as the map evolves. After each iteration, causal relations are updated according to the values obtained for each concept, which allows the map to "learn" the best values for each causal relation.

Unfortunately, none of the extensions discussed above enables us to model the kind of relation we identified in private information disclosure, namely, the relation between concepts and weights of causal relations. Our proposed model, FCM-CF, was designed to model the privacy impact of disclosing personal information, but can also be useful in other scenarios where the relation between concepts and causal relations must be captured.

5.4 Defining Disclosure Constraints for Personal Data

We propose a model based on FCM-CFs to enable individuals to measure the impact of disclosing personal information when executing composite services. Users first create an FCM-CF that represents the privacy impact of revealing each personal attribute individually. This definition does not need to be performed manually by each user: they can be based on the values defined by other users or suggested by privacy protection organizations such as EPIC[4] or the Electronic Frontier Foundation (EFF)[5].

After that, the middleware combines the FCM-CFs to model the effects of revealing subsets of data to the same entity. This final FCM-CF can be used to compare the privacy impacts of equivalent executable workflows featuring different service partitions or to simulate the effects on privacy when disclosing personal information with different levels of precision, as described in Sect. 5.5.

5.4.1 Characterizing Data Privacy Impact

There are two properties of private data that determine the privacy consequences of its disclosure: **identifiability** and **sensitivity**. The idea of using an attribute identifiability and sensitivity as a metric for the privacy impact of its disclosure is shared by the work in [IY04]. It proposes a model to automatically generate policies for attribute access control based, among others, on these two properties.

[4] epic.org

[5] www.eff.org

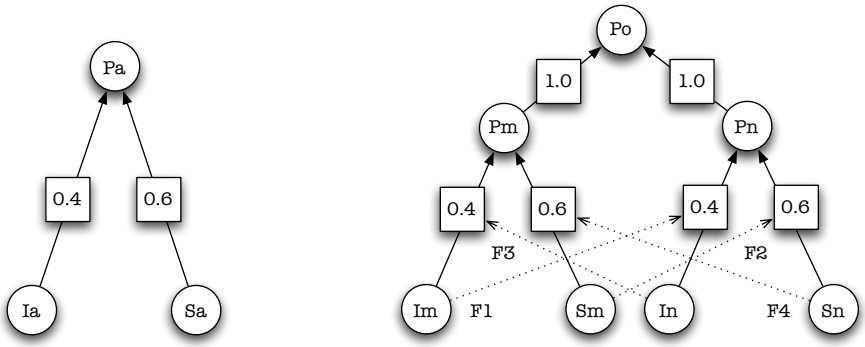
Identifiability measures how easy it is to identify an individual after disclosure of a specific information; it is a metric related to the distribution of an attribute a with value v on a population U. We consider two metrics of identifiability: how much a user can be typically identified by the attribute a (I_a) and how much a particular user can be identified by a specific value v of attribute a (I_v). Definition of the value for I_v requires knowledge about the distribution of v on the population. When this distribution is unknown, I_a can be used as an approximation of I_v. Those values are expressed in the interval [0,1] where 0 means that the attribute a or the value v are common to all members of the population U and do not allow to identify the user, and 1 means that the attribute a or the value v are exclusive to the user and can identify him uniquely. The value of I is the value of I_a if $I_v = 0$, or I_v otherwise.

Sensitivity, on the other hand, is a metric associated with the user perception of privacy. It measures how comfortable the user is when disclosing a particular attribute a with value v. We also consider two metrics for sensitivity, namely, the attribute sensitivity (S_a) that measures how sensible it is for the user to disclose attribute a in general, and the value sensitivity (S_v) that quantifies how embarrassed the user is to disclose value v of attribute a. Those values are also expressed in the interval [0,1] where 0 means that the user does not see any issue when revealing attribute a or value v, and 1 means that the user is very embarrassed to reveal attribute a or value v. The data sensitivity S is also defined as S_a if $S_v = 0$, or S_v otherwise.

In many cases private information can be disclosed using different precision levels. For instance, when asked for his age, the user can disclose his exact age (e.g., 27 years old) or just his age range (between 20 and 30 years old). Each possibility represents different degrees of identifiability and sensitivity on the interval $[0, 1]$. The values of I and S in this case may vary according to the precision of the information the user reveals. In other cases personal information can be decomposed into smaller components. As an example, a *complete name* contains a *first name* and a *last name*. The next section discusses how to combine information components to model the privacy impact on disclosing a composed information from the impact of disclosing its parts.

5.4.2 Modeling Privacy Impact

The privacy impact of disclosing a specific personal information can be obtained from its identifiability and from its sensitivity as defined previously. Since sensitivity is a user-related metric, we give it a bigger weight than identifiability so that the user's perspective on privacy prevails. The sum of their weights does not have to be 1, the values only represent that identifiability has a positive effect slightly smaller than average (0.4) and that sensibility has a positive effect a little bigger than average (0.6) on the privacy impact of disclosing the information. The FCM-CF representing the privacy disclosure impact of an atomic personal attribute is described by Fig. 5.16 (I), where Ia is the identifiability of information a, Sa is its sensitivity and Pa is the privacy impact when the information a is disclosed. If information a has different levels of detail – e.g., location can be expressed in GPS coordinates, name of the street or city name – this diagram can be used to measure



(a) (b)

Figure 5.16: An FCM-CF Modeling the Privacy Impact of Disclosing an Atomic Personal Information (a) and a Composed Personal Information (b)

the consequences of disclosing data using different resolutions. The values of Ia and Sa are fuzzy and may assume different degrees according to data resolution. For instance, identifiability of location in GPS coordinates may be closer to 1 while identifiability of location as a city name is much smaller. For each possible data resolution i corresponds a pair $\{Ia_i, Sa_i\}$. Variables Ia and Sa can be replaced by values Ia_i and Sa_i on the map to compute the privacy impact Pa_i of disclosing variable a with resolution i.

Personal information can also represent a composition of other atomic information. Users can define the identifiability and sensitivity for each atomic attribute and create an FCM-CF that represents the impact caused by disclosure of all the attributes. Figure 5.16 (II) shows the FCM-CF of information o, which is composed by atomic informations m and n. This map shows how causality feedback is used to represent the relation between disclosure of different pieces of personal information. If information m is independent from n, or in other words, if knowledge of m does not affect knowledge of n, edges F_i have value 0 and the effect of disclosing m and n together (or o) is the same as disclosing each data separately. However, in many cases revealing one personal information has consequences on further disclosures. For example, the privacy impact of disclosing a first name increases if the last name was already revealed to the same entity. Edges F_i can thus be used to represent the increase on privacy impact when both data are revealed. Their values are defined according to the strength of the relation between the information data types.

5.5 Comparing the Privacy Impact of Service Compositions

After creating maps for all relevant personal information that the user may disclose, those maps can be combined to measure the privacy impact of service compositions. This section explains how the model is instantiated according to available service compositions and how

the user can select the one that poses the smallest threat to his personal life.

5.5.1 A Model for Reasoning about Privacy of Service Compositions

In principle, a user wants to perform a complex task without compromising his privacy, or in other words without disclosing any personal information. However, in some situations the user may have to reveal some personal data to obtain a certain result. For instance, a user who wants to buy a book and comfortably receive it at home will have to eventually disclose his home address. Privacy protection involves a negotiation between the privacy impact of personal information disclosure and the benefits the individual obtains from disclosing that information.

Sometimes, though, service providers may abuse their relation with the customer and require additional data not related to the service goal for marketing purposes or simply to enrich their databases before selling them to interested companies or governmental agencies. For instance, some services require the user's telephone, address and age to send an e-card, or the user's social security number to post comments in news articles. Disclosure of personal data necessary to execute a task can be viewed as an acceptable risk, while transmission of unnecessary private data may be deemed inadmissible.

Another element that plays an important role in the individual's decision to reveal private data is his trust in the service provider. Users feel more secure revealing private information to providers they already know and trust than to unknown providers that they are unsure of how are going to use that data, or even worse, to a familiar provider that they distrust because it has a bad reputation or because it has already invaded his privacy in past interactions. This relation is specific to each service provider as the privacy impact of disclosing the same information to a trusted or to an untrusted provider is different. As such, it cannot be established a priori, and depends on the service composition being considered for execution.

To model the compromise of disclosing private information to obtain a composite service functionality, we introduce four concepts into the maps that model the privacy impact of disclosing a private information: **convenience**, **inconvenience**, **trust** and **distrust**. Those concepts are particular to a service composition, and can only be determined after a composition is defined because they depend on the service providers involved in the composition and on the relation between the inputs required and the outputs provided by the composition. Their values are fuzzy, and can range from 0 to 1 in accordance with the definition of fuzzy cognitive maps.

Convenience is opposed to privacy, so the more convenient to the user it is to disclose a personal information, the smaller will be the privacy impact of its disclosure. Convenience has, thus, a weakening effect on the privacy consequences of personal data disclosure. Its value is fuzzy, and it is related to the necessity of disclosing the information to obtain the desired output. A value of 1 represents that the information is extremely related to the service purpose and thus it is convenient for the user to disclose it. Conversely, inconvenience expresses how unsuitable it is to disclose that information to obtain the desired goal. It increases the privacy impact of disclosing the information since its disclosure is

not required for the composition goal. A value of 1 means that disclosure of the requested information is highly inconvenient to the user, or in other words, that the information is unrelated to the service's goal.

Those values can be determined according to user experience (how many times that information was already requested to obtain a similar output), recommendation (privacy activist groups may publish lists of abusive requests) or legislation (defining which categories of private data that merchants are allowed to require when providing a specific output). It is not necessary to define both values since they are complementary. They could even be represented by a single concept with values in the range $[-1, 1]$, but this goes against the definition of FCMs.

Trust, as convenience, also has a weakening effect on the privacy impact of disclosing personal information. Its value is fuzzy because trust can be expressed in degrees, and a value of 1 represents that the user fully trusts the provider to adequately use his private data. Conversely, distrust is used to model situations where the user knows that a provider maliciously uses collected personal data. Distrust, thus, increases the privacy impact of disclosing an information since the user has reasons to believe that it will be misused, and a value of 1 means that the user is sure that the provider is malicious.

Trust and distrust are derived from many factors, such as the provider's reputation according to the opinions of other users, the existence and contents of a privacy agreement and the past experiences of the user with that service provider. Trust management systems such as [Cap04, LI04b] can be used to determine the trust values of each service provider. Falcone *et al.* [FPC03] propose a model that uses Fuzzy Cognitive Maps to compute trust from multiple factors such as the provider's ability of producing a certain output, its availability and the risk (or danger) of using the provider. Their approach can be used to determine the user degree of trust on a service provider and be combined to ours to establish the privacy impact of disclosing data to that service provider. Other models of trust can also be integrated to the map that represents the privacy impact of disclosing multiple personal information to the same provider. As it happens with convenience/inconvenience, our model does not require the definition of values for both trust and distrust. Those concepts are complementary and are not represented by a single concept in the range $[-1, 1]$ to respect the FCM framework.

Figure 5.17 shows an FCM-CF representing a service composition that requires three types of personal data. To model the privacy impact of the composition, a new concept P representing the total impact of executing the composition is created. The particular privacy impact of every information required to run the composition is connected to P with weight value 1. This is due to the fact that each information contributes equally to the privacy impact of their combined disclosure. Moreover, the effect of their disclosure is not attenuated when they are revealed during the execution of a single service composition or when executing each service separately.

Each information has its specific convenience factor C, that has an effect of -0.4 over the privacy impact of its disclosure. This weight means that the necessity of disclosing a specific information to obtain a desired output has a negative effect below average on the privacy impact of executing a composition. This concept could be replaced by a C^- concept

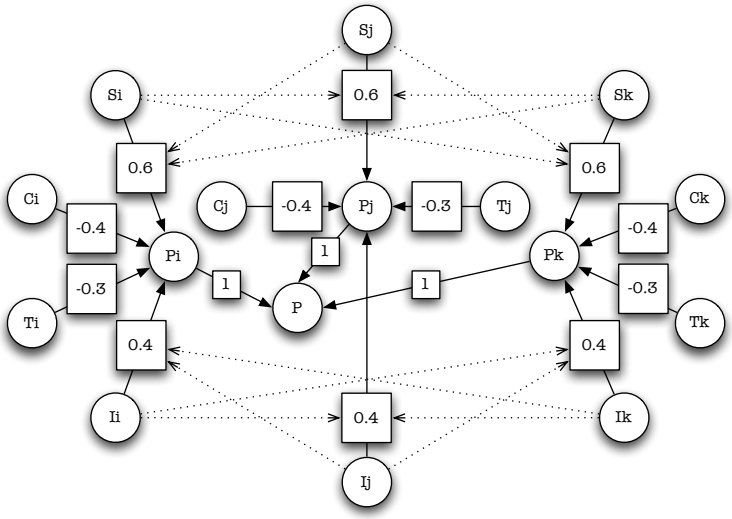


Figure 5.17: An FCM-CF of a Composition Requiring Three Atomic Personal Data

with effect 0.4 to represent the complementary concept of inconvenience. The map of each information also contains the trust that the user has on the service provider that receives the information. It has an effect of -0.3 over the information privacy impact, which corresponds to the idea that disclosing private information to a trusted service provider slightly reduces the privacy impact of its disclosure. We consider a small impact reduction because even though the user may trust a provider to fairly use his information, attacks from hackers or company merges can affect usage of personal information already disclosed to a trusted provider. This concept could be replaced by a concept T^- to represent the user distrust on the service provider, with causality effect of 0.3.

Finally, the model requires weights for all dashed lines representing causality feedback. Identifiability of an information can only affect the identifiability of another information, and sensitivity likewise. In service composition, causality feedback happens when related personal information is disclosed to the same service provider. When data is disclosed to different service providers, the composition does not present causality feedback and the weight of all dashed arrows is 0. However, if the composition requires sending different information to the same provider, the CF edges relating those informations will have weights proportional to the effect that disclosure of one information has on the privacy impact of disclosing another.

The causality feedback between two personal data is positive when they reveal more private information about the user together than separate. For instance, when the zip code and the gender of an individual are disclosed to the same entity, we can say that knowledge of the zip code greatly increases the identifiability of gender, while knowledge of the gender slightly increases the identifiability of zip code. Causality feedback between two data can also be negative in the case where one information helps to contextualize

the other, and disclosure of both to the same provider causes less impact then revealing only one of them, which could cause incorrect interpretations. For instance, a professor of European History buying books about the Nazism may want to reveal his profession to avoid misinterpretation of his intentions.

Some service discovery protocols explicitly provide mechanisms for service provider description (such as the *bussinessEntity* field in UDDI [SH05]) and the model can use this information to define which causality feedback edges are active when evaluating a specific workflow. Other discovery protocols do not particularly support service provider specification, but this information could be extracted from data such as the service access point URL or external directories that categorize service providers.

Since only a few causality feedback edges are used to model the privacy impact of disclosing groups of personal data to the same provider, the resulting CF graph is very sparse which reduces the complexity of computing FCM-CF states. Also, since the resulting FCM-CF does not contain cycles, the map stabilizes after a few iterations. The value of outermost concepts never changes among iterations, so the edge values stabilize after a few iterations. Since there is no feedback, the concept values also stabilize after a small number of iterations.

5.6 Illustration of the Model Usage

To present how our model can be used to evaluate the privacy impact of executing different service compositions we introduce in this section a pervasive computing service composition scenario that can be realized by two possible executable workflows and show how the model described in previous sections can help the user to determine the privacy impact of each workflow and execute the one that causes a smaller impact on his privacy.

William is passing his vacations at a city very distant from where he lives. Even though he suffers from a critical medical condition, due to advances in treatment he can lead an almost normal life. This morning, however, he is not feeling very well and he thinks it would be better to see a doctor. From his hotel room, he uses his smart phone to book a taxi that will take him to the nearest hospital that has a treatment center specialized on his condition. He uses a pervasive web service application provided by his personal doctor to help him to perform this task.

The abstract composition workflow used by the application is depicted on Figure 5.18. It consists of three tasks: the first tries to locate the user based on local information such as cellular phone towers and wireless hotspots nearby. The second task receives as input the user location and a type of business (in this case an hospital with a specific treatment center) and outputs the address of the closest establishment. Finally, the last task books a taxi based on the address of origin and destination, the time for the trip, and a phone number for confirmation.

Figure 5.19 shows two possible executable workflows that provide the user-required functionality but present different privacy properties. Tasks inside the same capsule are executed by the same provider. Workflow I shows a situation where all the services are

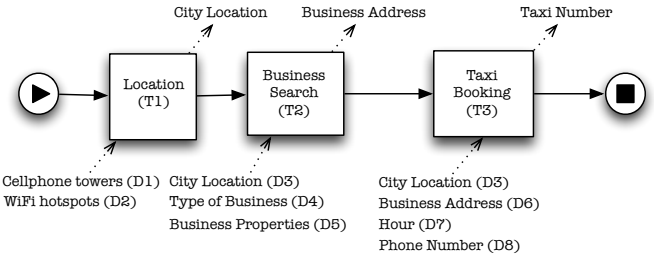
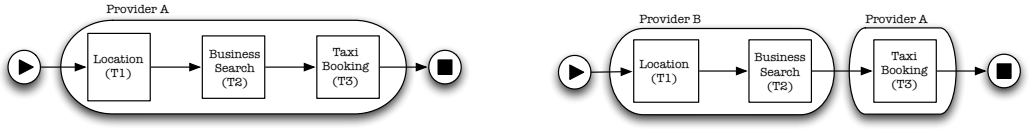


Figure 5.18: Abstract Workflow of a Service Composition



(a) Executable Workflow I

(b) Executable Workflow II

Figure 5.19: Possible Executable Workflows of the Example Abstract Workflow

offered by the same entity, while workflow II represents an executable workflow where location-aware search is performed by one service provider and taxi reservation by another.

In our example we consider four situations. We use the model to compute the privacy impact of executing the composition depicted in workflow I by an unknown provider (whose trust we consider to be 0) and by a moderately trusted provider (trust equal to 0.5). We also compute the privacy impact of executing the workflow II when both providers are unknown and when both of them are moderately trusted.

5.6.1 Building the Model

Two steps are necessary to model the privacy impact of composite service execution. First the user must create the model, and then instantiate it with information specific to the workflow. Model creation is independent from the executable workflows and can be performed offline. It consists of defining the identifiability and sensitivity of each personal information that can be provided by the user, and the causality feedbacks that appear when different data is disclosed to the same entity. In this example, we will focus on the eight types of data required to execute the composition. Typical values are used for most of the data, except for the values "hospital" and the specific treatment center that have user-defined identifiability *medium* and *high*, and sensitivity *high* and *very high*. Typical values can be obtained from other users or from privacy activism groups, and can be related to particular inputs on service composition workflows by using ontologies (for instance to identify that *city location* is a more specific type of *location*). Table 5.1 contains the values assigned to identifiability and sensitivity of each data.

Table 5.1: Definition of the Identifiability and Sensitivity of each Personal Data

Data	I_{a_i}	I_{v_i}	S_{a_i}	S_{v_i}
D_1	0.2	-	0.4	-
D_2	0.2	-	0.4	-
D_3	0.3	-	0.5	-
D_4	0.4	0.5	0.5	0.8
D_5	0.4	0.7	0.5	1.0
D_6	0.3	-	0.6	-
D_7	0.3	-	0.6	-
D_8	0.8	-	0.5	-

The weight of causality feedback edges can also be defined in advance. Afterwards, depending on the executable workflow under evaluation, those edges may receive the value 0 (for data disclosed to different entities) or the pre-defined value (for data disclosed to the same entity). Considering our scenario above, disclosure of the phone number greatly increases the identifiability of other personal data disclosed to the same entity. We define CF edges with value 0.7 from concept D_8 to the weights of all edges connecting data identifiability with its privacy impact.

5.6.2 Using the Model

The model above must be instantiated for each executable workflow under consideration. Model instantiation consists of three steps: (1) definition of convenience values C_i for each information according to the need to disclose it to obtain the desired output, (2) definition of trust values T_i for each information according to the trust that the client has on the provider accessing that information and (3) neutralization of CF edges connecting data that is not disclosed to the same entity on the executable workflow. In this example, every information required by the tasks is necessary to obtain the desired composition output, so all the convenience values are set to 1 in all model instances. The trust on the service provider varies according to the scenario: when the provider is unknown, the value of trust is 0 and when it is moderately trusted, the value is 0.5. Causality feedback edges are also different according to the scenario: the models for the first executable workflow contain all causality feedback edges since the same provider has access to the user's phone number and all other data the user provides, while the models for the second executable workflow only have CF edges between the identifiability I_8 of data D_8 and the identifiability of other data accessed by the same entity, namely I_3, I_6 and I_7. Figures 5.20 to 5.23 show model

instances for all four cases.

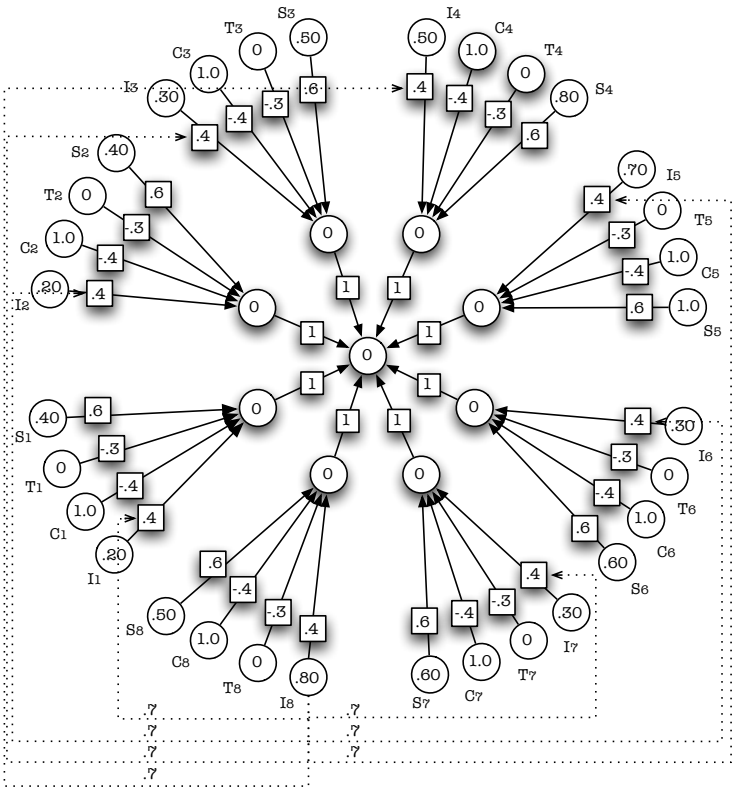


Figure 5.20: Model for Executable Workflow I and an Unknown Provider

Once values for all concepts are defined, we can iterate the FCM-CFs until they reach a stable state to compare the privacy impact of each executable workflow. The iteration begins by computing the new weights of all causality edges that are influenced by CF edges as explained in Sect. 5.3. In models 5.20 and 5.21 this results in changing the weight of all edges connecting the identifiability of an information to its privacy impact from 0.4 to 0.74 after the first iteration. In models 5.22 and 5.23 only edges coming out from concepts I_3, I_6 and I_7 suffer causality feedback, so only their weights are updated to 0.74 after the first iteration, while the others remain at 0.4.

After updating the weights, the new state of the FCM-CF can be computed as in traditional FCMs. This process is repeated until both the CF and the FCM graphs become stable or generate a cycle. Those maps are feed forward – or in other words, they do not contain cycles, or feedbacks – which greatly simplifies next step computation. All maps reach a stable state after approximately 25 iterations, and the final value of edges suffering causality feedback is 0.897 in all maps. Final values of the central concept P which represents the composition privacy impact for all four cases are given in Table 5.2.

As we can see from Table 5.2, the executable workflow that produces the smallest

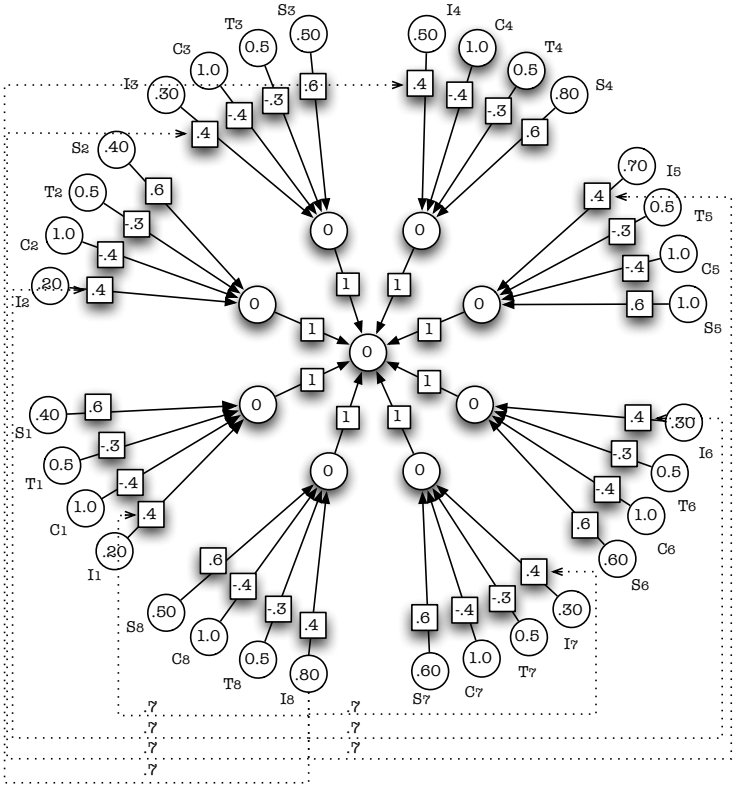


Figure 5.21: Model for Executable Workflow I and a Moderately Trusted Provider

impact on the user privacy is the one depicted in Fig. 5.19(b) when both providers are moderately trusted by the user. Indeed, by disclosing information that together can be potentially privacy invasive to different service providers, the user can reduce the privacy impact of data correlation. Moreover, since the user averagely trusts the providers to fairly use his private data, the impact is smaller than using an unknown provider that may maliciously use the data.

When considering only unknown service providers, or those whose trust the user could not establish either because it is their first interaction or because the user still does not have enough information to determine how much he trusts the provider, it is still better for the user's privacy to execute the workflow in Figure 5.19(b) than the one in Figure 5.19(a). This is because providing information that becomes more sensible when analyzed together to a single untrusted provider is more dangerous than splitting that information to different providers. However, since the user considers both providers as untrusted in the case of Figure 5.19(b), there is a possibility that his private data can be misused, which means that eventually the providers can exchange that information and correlate the data. As we can see in Table 5.2, the privacy impact of the map in Figure 5.22 is smaller than the one in Figure 5.20 but both are still close to 1, which means that both situations are highly

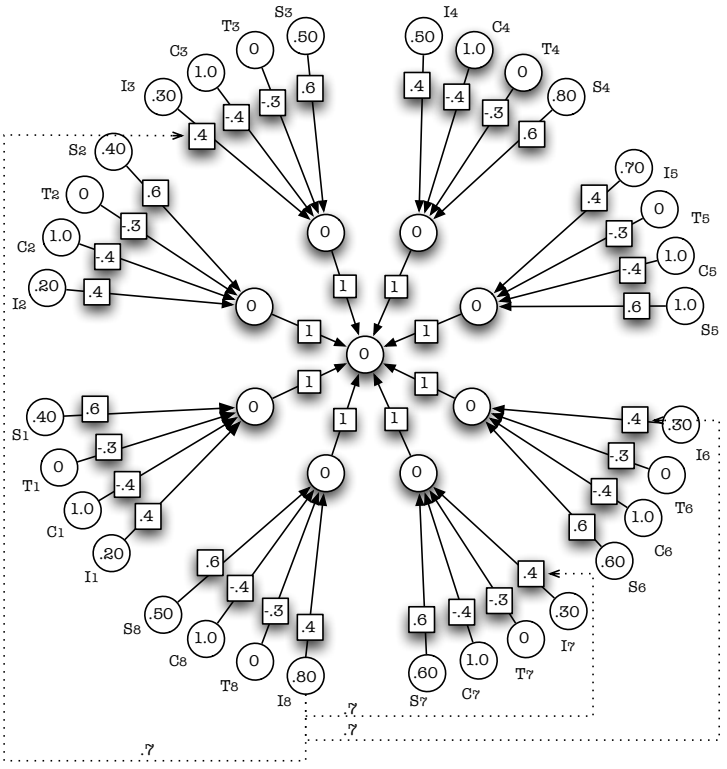


Figure 5.22: Model for Executable Workflow II and Unknown Providers

dangerous for the user privacy.

5.7 Model Assessment

One important issue with fuzzy cognitive maps is that it is computationally hard to identify if a given map will stabilize after a number of iterations. Negative feedbacks and cycles can keep concept values constantly changing, and the map may not stabilize even after infinite iterations. It is also hard to detect stability: if concept values keep changing but repeating the same cycle of values, the map is considered to be stable. However, there is no limitation on the period of such cycles, and a potentially large number of map states must be stored to enable detection of a cycle of states.

The maps created from the model described in this chapter, however, do not present this problems. By construction, maps that model the privacy impact of disclosing personal information do not contain feedback edges: all edges are directed to the central concept that represents the privacy impact of the composition. In such situations fuzzy cognitive maps are known to stabilize since the concepts that generate causality have constant values. Moreover, as the maps are feed forward with constant values for concepts creating

Table 5.2: Privacy Impact of each Executable Workflow

Executable Workflow	Privacy Impact
Single provider, unknown (5.20)	0.966
Single provider, moderately trusted (5.21)	0.833
Two providers, unknown (5.22)	0.914
Two providers, both moderately trusted (5.23)	0.601

causality, the map states do not enter a cycle. Fuzzy cognitive maps created from this model stabilize after a few iterations.

Limitations and Extensions

One of the limitations of the approach proposed in this chapter is the way it handles black-box composite services. From the user point of view, those services are just common atomic services provided by the entity that announces them. In reality, however, they are a composition of services supplied by multiple service providers. Composite service execution, in this case, is performed transparently on behalf of the user: the client provides all inputs to the composite service, which orchestrates the invocation of services that are part of the composition. Since clients do not know which services are invoked, they cannot correctly evaluate the privacy impact of executing the black-box service.

This problem can be solved using two complementary techniques. The first is to require that services publish their conversations as well as their interfaces. This way, a user who orchestrates the execution of a service composition that contains composite services itself can use each composite service conversation to identify other service providers and to reason about the possible correlations that may happen when executing the whole composition. Users can then assign higher trust values to providers that publish their conversations and lower trust values for providers that do not publish it.

Another technique that can enforce that data transmitted to a service provider will not be forwarded to unauthorized providers is data usage control [PHB06]. Whenever a user invokes a service he can protect the data with usage control mechanisms to ensure that only the service he is invoking is able to read the data. This way, if a service that the user imagines to be atomic is actually a black-box composite service, the invocation will fail but user personal data will not be disclosed to service providers that were not accounted for when reasoning about the composite service execution privacy impact.

A further issue that is not currently supported by the model proposed here is the relation between consecutive service executions. As time passes, personal data that the user disclosed in a past interaction with a service provider may be related to personal data the user reveals in a further invocation. For instance, imagine that the user discloses his

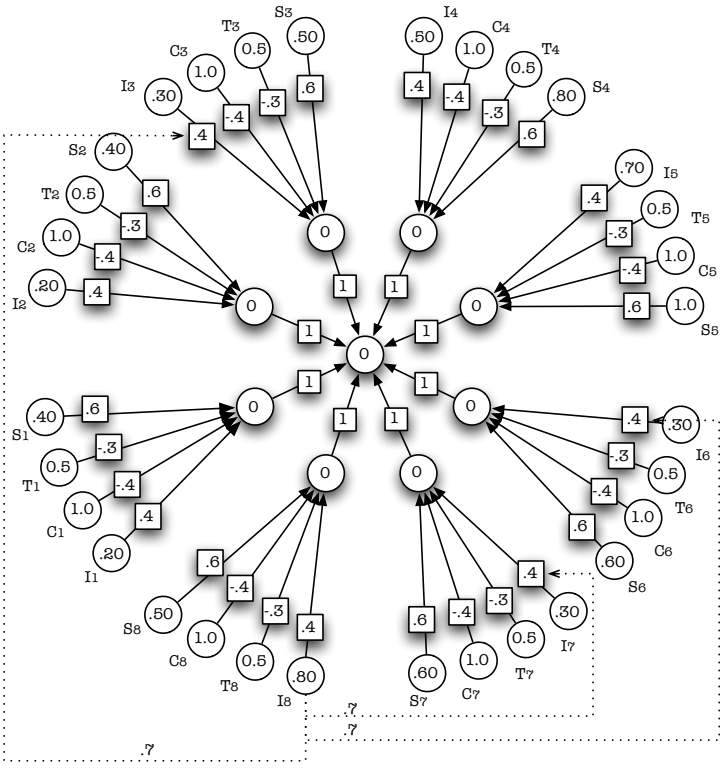


Figure 5.23: Model for Executable Workflow II and Moderately Trusted Providers

address to a service provider in an invocation and that he discloses his name to the same provider in a later invocation. If those calls can be correlated by the service provider (for instance through a customer number or the client phone number) the privacy impact of executing one service and than the other is similar to the impact of executing a service composition that contains both services.

The privacy impact increase resulting from successive disclosure of private information, however, does not happen in every disclosure nor with every data. If the provider cannot relate consecutive data disclosures, it does not gather more knowledge about the user and the resulting privacy impact is the equivalent of disclosing data to different providers. Other personal data, like home address, lose meaning as time passes. Disclosure of a former home address has a smaller effect on privacy than disclosure of the user current home address. Depending on the data disclosed and on the time between disclosures, the privacy impact can be reduced.

The model presented in this chapter could be extended to include the notion of time. The map used to compute the privacy impact of executing a service composition could be extended to take into account data disclosed to the same service providers on the past, and thus consider the potential for service provider data correlation of consecutive service calls. Moreover, these time-aware FCMs could implement the notion of aging,

using different values for identifiability and sensitivity of an information already disclosed as time passes. As a result, the privacy impact of executing a service composition would depend not only on the data involved in the composition but also on other contextual data such as past user iterations and the time elapsed among them.

Finally, Fuzzy Cognitive Maps may not be the most appropriated model to support those extensions. To create the model described in this chapter we already came across a number of limitations of FCMs, which lead us to even propose an extension to this model. Before creating those extensions, we plan to investigate Neural Networks [Kos91] and how they can be applied to this problem. Neural Networks are more flexible but it is also harder to create a stable model that corresponds to real situations.

"Could ya please not stare at me like that", he said,
"It's just my foolish pride,
But sometimes a man must be alone
And this is no place to hide."

Bob Dylan, "The Ballad of Frankie Lee and Judas Priest"

6

A Privacy-enhanced Service-Oriented Middleware for Pervasive Computing

In Section 1.2.1 we motivate the need for a privacy-enhanced middleware in pervasive computing. In summary, some assumptions of middleware developed for traditional distributed systems do not hold in pervasive: the network infrastructure is no longer trusted, protocols that require a trusted third-party have limited relevancy since connections are intermittent and third-parties may not be reachable, and enforcement of data usage rules is complicated since nodes move and are untrusted. Those issues have been neglected through middleware evolution and even though many initiatives for mobile computing or pervasive computing middleware exist, just a few consider the privacy impact caused by the middleware itself. This is particularly dangerous in pervasive computing, since the unobtrusive aspect of user interactions in such environments together with their presence in everyday activities leads users to disclose more personal information then they would do when interacting with classic single-purpose distributed systems.

For this reason, middleware for pervasive computing must consider privacy issues and must present additional features to reduce its own impact over the privacy of its users. In this chapter, we present PREMISE, a PRivacy-Enhanced MIddleware for SErvice-oriented

pervasive computing. PREMISE integrates the mechanisms for privacy protection during service access, discovery and composition presented in the previous chapters to extend an existing service-oriented middleware for pervasive computing. PREMISE is an extension of PLASTIC, a middleware that facilitates creation and deployment of mobile services in pervasive computing.

In the next sections we begin giving a brief overview of PLASTIC in Section 6.1, a service-oriented middleware for pervasive computing that we improved with privacy-aware and privacy-enhanced components. After that, in Section 6.2, we present the architecture of PREMISE detailing the middleware components executed by regular resource-constrained nodes on the pervasive environment and the additional components that are executed by more resourceful nodes willing to act as bridges and forward traffic between networks, or to act as service directories to receive service announcements and discovery requests. After presenting the architecture, we explain how the middleware components interact to enable local and remote provision and consumption of pervasive services with improved privacy protection. We introduce the scenario in Section 6.3 and then we describe how PREMISE components cooperate to increase privacy when the user defines a service composition (Section 6.3.1), discovers available services (Section 6.3.2), selects one composition from the multiple compositions available (Section 6.3.3) and finally accesses the composition to accomplish a task (Section 6.3.4).

6.1 PLASTIC Overview

The PLASTIC[1] project aimed at developing a comprehensive provisioning platform for software services deployed over multi-radio networks. The project builds upon both Web services and standard component-based technologies and integrates methods and tools for service development, from design to validation, and a supporting middleware for service provisioning in multi-radio networks. In particular, the main objectives for the PLASTIC middleware are (1) allowing the deployment of services over a large diversity of terminals, including (mobile) wireless, resource-constrained ones and (2) supporting advanced functionalities for mobile adaptable services, i.e.: context-aware service management, trust and security management, SLA enforcement, and information dissemination.

Towards the first objective, the PLASTIC middleware (depicted in Figure 6.1) builds upon the Web Service Architecture, so as to benefit from the pervasive nature of Web technologies that makes them available in most digital environments (*Multi-radio SOAP layer*). The PLASTIC middleware assumes an all-IP environment but without global routing, and enables the effective exploitation of multi-radio networking capabilities by composing the various networks in reach to improve availability of services and to offer seamless mobility. This specifically calls for routing protocols for the multi-radio network (*Multi-network routing*), where routing should meet requirements associated with both the communication protocols used at the application level and the features of the underlying composed networks (*Multi-radio unicast* and *Multi-radio multicast*). Also, the middleware

[1] www.ist-plastic.org

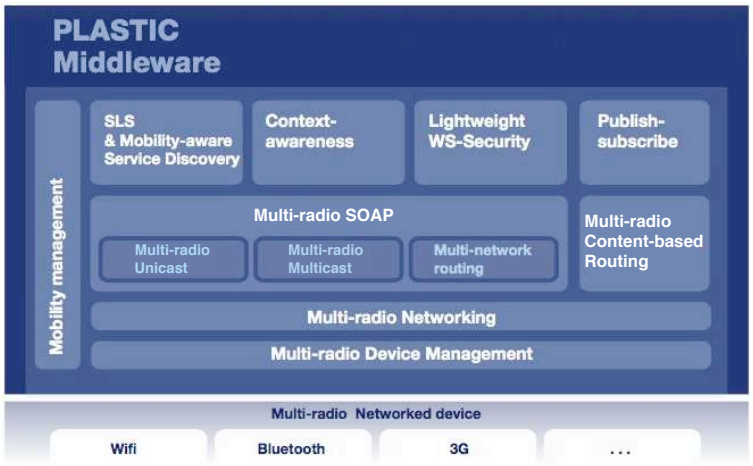


Figure 6.1: The PLASTIC Middleware

manages the various radio interfaces (*Multi-radio Device Management*) that are embedded on wireless devices, offering the abstraction of an integrated multi-radio interface to the software services of the upper layers (*Multi-radio networking*). Such abstraction increases the quality of service access by exploiting the underlying redundancy of network connectivity that results from having distinct radio network links directly connecting two nodes in a transparent way for users.

With respect to the work presented in this thesis, it is important to understand how PLASTIC handles user mobility and network interface diversity during service provision and consumption. In order to identify applications running on the network, PLASTIC assigns to each application a unique PLASTIC-Address. This address remains fixed regardless of the network interface used to access the application or the IP addresses used by the interfaces [RFIG07]. The PLASTIC-Address is automatically generated and managed by the *Multi-radio Networking* component, and resolves into the actual set of IP addresses bound to the device hosting the application. Upper layers can use this address instead of the traditional IP-based addressing scheme to communicate with the application in spite of connectivity changes.

The Multi-radio Networking layer provides two types of communication facilities: (*i*) synchronous unicast is used to read/write packets exchanged during the interaction between client and user applications, and (*ii*) asynchronous multicast allows the user application to send multicast packets to all members of a group. Finally, this layer also activates and selects the best possible network interfaces (among those available) with respect to application- and user-required QoS level. On the top of this layer, the *Multi-radio SOAP* component provides SOAP communication through multiple networks using PLASTIC-Addresses. It also uses the communication primitives that the *Multi-radio Networking* component supplies to provide unicast and multicast SOAP communication primitives [C+08].

PLASTIC also introduces a language for service descriptions called PLASTIC Service Description Language (PSDL). Instead of being yet a new language for service descriptions, PSDL serves as a container for multiple documents describing different aspects of the service using standard languages. For instance, the service interface can be described using WSDL, the description of its non-functional properties such as QoS can use specific languages such as SLAng [LSE03] and its conversation can be defined using BPEL. PSDL thus allows for a more complete description of a service without creating a new non-standard description language. Legacy services described using only one of those aspects (only the WSDL interface, for instance) can be easily transformed into a PLASTIC Service Description (PSD) by adding a reference to the WSDL document into the interface section of the PSD file.

6.2 The PREMISE Middleware Architecture

As explained in Section 1.2.4, we assume an environment composed by independently managed networks without global IP addressing, which we call HMANETs. Routing among different networks is provided by bridges, devices simultaneously connected to more than one network that voluntarily forward packets on behalf of other users. Bridges enable users to discover and access services in remote networks, augmenting the set of available services and allowing users to cooperate even if they are not connected by the same network.

These characteristics of HMANETs lead us to adopt a modular design to PREMISE, with components that can be optionally instantiated according to the role that the mobile device plays in the environment (client/service provider or bridge/service directory) and the quantity of available resources. This approach helps to reduce the middleware load on resource-constrained devices that only need, for instance, to consume pervasive services. Figure 6.2 shows the PREMISE configuration for a regular node.

A regular node is one that consumes or provides services to other nodes in the local network. A regular node can play both roles at the same time, and may be connected to more than one network using heterogeneous technologies. In this case, however, the regular node does not forward packets from one of the networks to which it is connected to another. A regular node that consumes services must be able to discover them, select the best service among those found and finally access the service. A regular node providing services needs only mechanisms to publish service descriptions and to answer service requests.

Figure 6.2 highlights the PREMISE middleware components that are added to the original PLASTIC architecture on regular nodes, to improve privacy during service access, service discovery and service composition. Nodes only acting as service providers do not need to instantiate the service composition module and can thus reduce the middleware footprint. This privacy protection layer provided by PREMISE uses the communication primitives provided by the PLASTIC middleware to communicate in heterogeneous networks, and provides to the upper layers service-oriented functionalities enhanced with privacy protection. The details of each component and their interactions are presented in

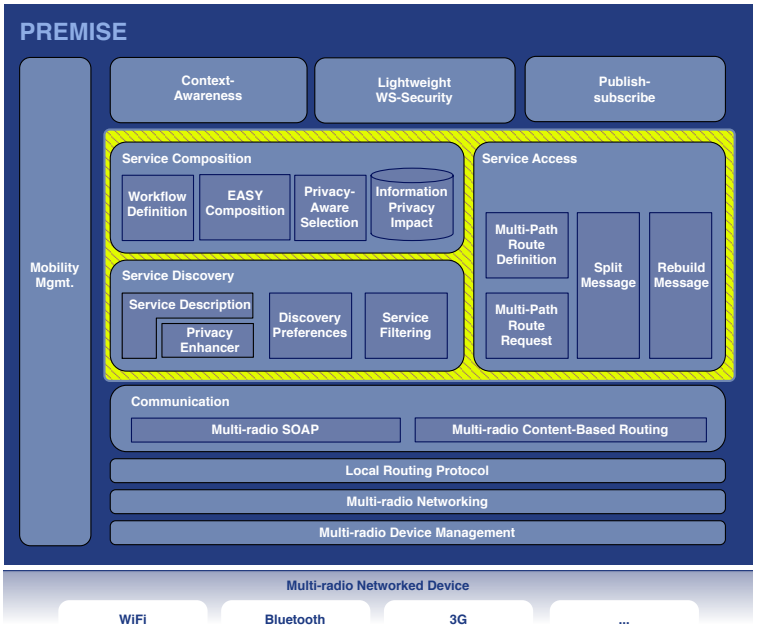


Figure 6.2: PREMISE - Regular Node

the following sections.

Regular nodes are unaware of connections to remote networks and inter-network routing. Those nodes use their local network routing protocol to access local service directories and services available nearby, and thus do not need to instantiate components that handle overlay routing. Every routing operation involving remote networks is performed by the local service directory on behalf of the node. Regular nodes also do not forward traffic to remote networks, so they do not need to instantiate any component dedicated to improve privacy protection when forwarding messages.

Figure 6.3 shows the middleware components that a node willing to act as a bridge must instantiate. Such nodes can have two roles: they can be connected to different networks and forward packets among them working as bridges, and they can serve as a service directory for the local network handling service announcements and requests. If the node is only working as a service directory, it must only load the *Service Directory* and the *EVEY Privacy-Enhanced Matcher*. If it is also working as a bridge, it must route packets among different networks and thus must execute all components responsible for maintaining the organization of the overlay network plus the components that increase privacy protection when forwarding messages to remote networks. Privacy protection mechanisms are different for service discovery messages and service access messages, so they are handled by different components as detailed in Sections 6.3.2 and 6.3.4.

Bridges can consume or provide services while forwarding messages, in which case they must also instantiate the same components as regular nodes to have additional privacy

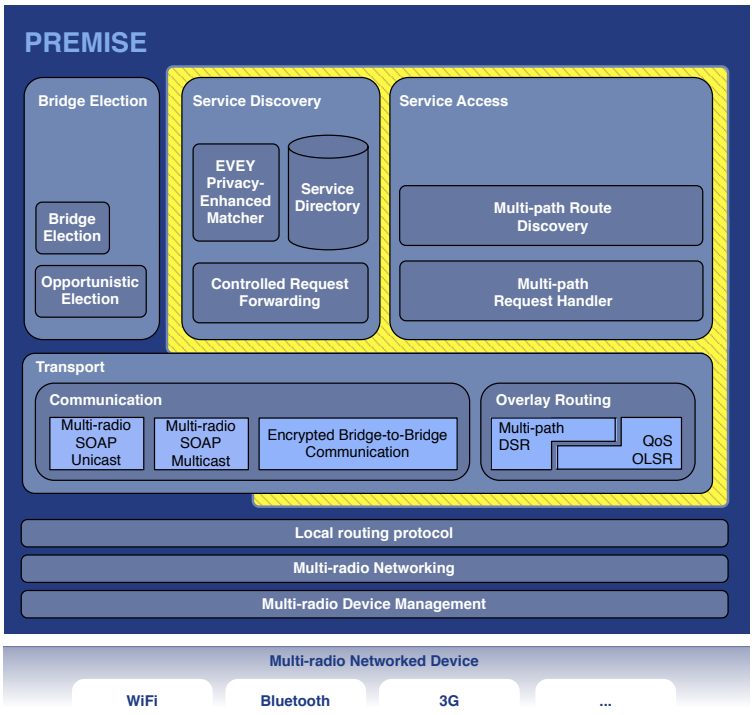


Figure 6.3: PREMISE - Bridge

protection during service discovery, composition and access. On the other hand, a node acting as a pure bridge or service directory does not consume services so it does not have to run components that provide privacy protection during service composition. Additionally, a bridge does not have to generate privacy-aware service descriptions and does not need to run related components; matching between discovery requests and service advertisements is performed using privacy-aware versions of both documents.

In the next section we introduce a sample application that motivates the middleware design and the privacy protection components of PREMISE. Through this scenario, we show how middleware components cooperate to increase privacy of service provision in pervasive computing. Based on this scenario, we explain the middleware features for privacy protection during service access, discovery and composition.

6.3 The "Cupd" Application Scenario

John is single and is member of an online dating community called Cupd. Although he is a big fan of the service, which allows him to meet interesting people that he would not meet otherwise, he prefers not to discuss about it with his friends. He believes that his group of friends have a pre-conceived bad idea of this type of service and he is afraid of

his friends' judgement if they discover that he uses one such service.

John is spending his summer vacation in a city by the sea, and while relaxing at the beach he wonders if any of the girls around him is also single and interested in meeting new people. Cupd has created this convenient mobile service that enables a user to check if any other member of the Cupd community is around and if their profiles are compatible. He sends his profile to the matching services he finds around him and receives a few positive answers containing the pseudonym of other Cupd users.

After reviewing the compatible profiles, John decides to directly contact one of the users. Whenever a Cupd member wants to directly contact another member, they must communicate using the tools provided by Cupd. This is a particularly attractive feature of the community since it allows for moderation of offensive language, protection against harassment, creation of blacklists and other mechanisms for managing communication with other Cupd members. John uses the Cupd communication system to invite his date to a dinner that same evening.

Some time later, John receives a message: his invitation was accepted! He decides to make a reservation on a fancy Thai restaurant to make a good impression on his date. He sends the reservation request and in a couple of minutes he receives a call from the restaurant to confirm his table. With everything set for the evening, John picks up his mask and goes out for some scuba-diving.

Technical details John is looking for three services. The first is a mobile ad hoc service that is announced to users nearby and that performs a profile match. John sends his Cupd profile to the user running the service, and if his profile matches the profile of the person providing the service, John receives that profile for evaluation. This service is deployed over the ad hoc network enabling John to perform location-based profile matches with people nearby without continuously disclosing his location to a central server, which would be both inefficient and undesirable with regards to privacy.

The second service is a communication service that allows John to send a direct message to a Cupd user and to receive the user's answer. This service is hosted by a remote server controlled by the company providing the Cupd service. This is necessary to enable moderation and control of the communication between any two members. We assume that this service is deployed on an infrastructure such as the telephone network or the Internet and that users are able to access it through an SMS or a GPRS connection.

Finally, the third service takes as inputs a date, John's name, the number of persons invited and his phone number for confirmation, and provides him a reservation number for a table in a restaurant. We consider that there are two versions of this service available: one provided by the city hall that lists all restaurants available in the city and enables tourists and local residents to book a table, and another provided by Cupd that gives price reductions on a number of partner restaurants for reservations resulting from meetings organized through the community. Figure 6.4 shows the service interfaces in UML and Figure 6.5 shows the abstract service composition using the same notation introduced in Chapter 5.

We assume that John's mobile phone has multiple network interfaces and he is able to

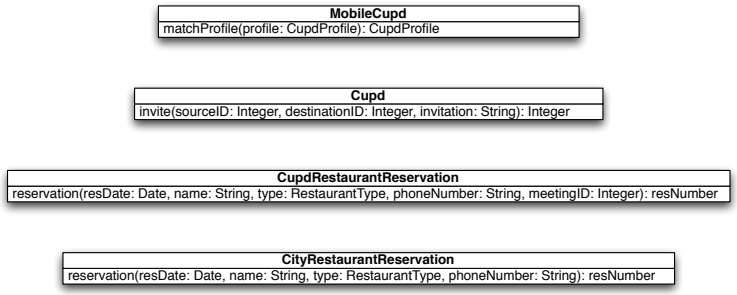


Figure 6.4: UML Diagram of the Services Described in the Scenario

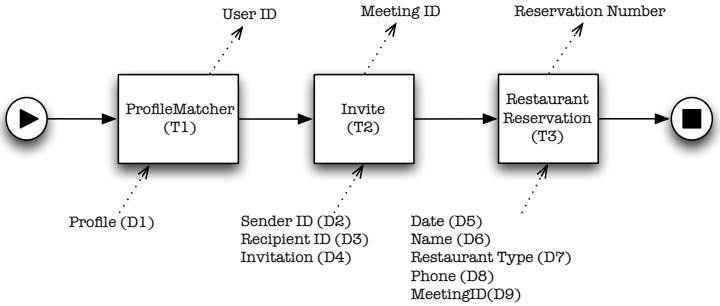


Figure 6.5: Abstract Workflow of the Scenario Composition

form ad hoc networks with other persons in the same place as him, for instance, the same beach. In the next sections we explain how middleware components interact to enable John to look for those services without disclosing information about himself, to choose the composition that better respects his privacy, and to finally access the service protecting his personal data from untrusted intermediary nodes.

Figure 6.6 shows the network configuration when John starts to look for a partner. Each network has a service directory (SD) responsible for managing service announcements and requests for its local network. Networks are connected to each other by bridges (B) and can either use ad hoc or infrastructure connections.

6.3.1 Service Composition

John is using an application created by a Cupd member. This application enables users to look for a partner, send an invitation and book a restaurant based on the partner's reply. The application is actually a service composition that combines Cupd services and the interface of a generic restaurant reservation service. John does not need to create the composite service himself: the user-created application contains the composition definition in a PSD document that contains a reference to a BPEL document specifying the service

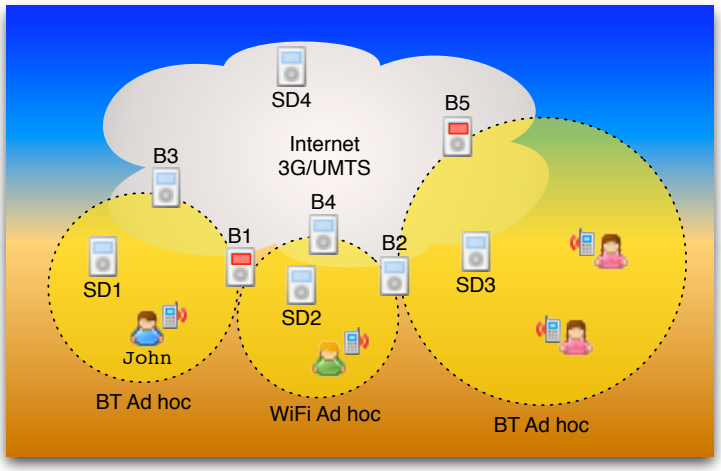


Figure 6.6: Network Configuration during Service Access

conversation and another reference to a WSDL file that specifies the composite service interface.

The middleware receives the composite service specification and looks for a suitable service in the environment. The middleware looks for services providing the same interface as the composite service and also for services that can be composed to provide the same functionality described by the service composition. The EASY [MPG+08] conversation-driven composition component that is part of PREMISE is able to use the composite service conversation as a basis to find available services that can be recomposed to provide the user-defined conversation.

Before starting to look for services to execute a conversation, John must define the identifiability and sensitivity of each personal information that can be revealed while executing the composition. Identifiability and sensitivity values of a personal information are the same regardless of the composition, so John may have already defined those values some time before. Also, John does not have to define these values manually for every possible personal data provided. Privacy activism organizations such as the Electronic Privacy Information Center[2] (EPIC) or the Electronic Frontier Foundation[3] (EFF) can publish lists with typical values for some commonly provided personal data that John can automatically import. The middleware can also provide an interface that asks John those values whenever data with undefined sensitivity or identifiability is required. Those values are stored in the *Information Privacy Impact* component of PREMISE.

John uses typical values for most of the information he needs to disclose to perform his task, except for his name. He increased the name identifiability because he has a Basque last name that is not very popular in his country and can thus be easily used to identify

[2]www.epic.org
[3]www.eff.org

Table 6.1: Definition of the Identifiability and Sensitivity of Personal Data

Data	I_{a_i}	I_{v_i}	S_{a_i}	S_{v_i}
D_1	0.5	-	0.9	-
D_2	0.3	-	0.3	-
D_3	0.1	-	0.5	-
D_4	0.6	-	0.8	-
D_5	0.1	-	0.1	-
D_6	0.9	1.0	0.6	-
D_7	0.1	-	0.2	-
D_8	0.9	-	0.5	-
D_9	0.0	-	0.0	-

him. Information D_9 corresponds to the `meetingID` generated by Cupd whenever an invitation for a meeting is accepted. Since this data is not directly related to John and he does not know what it represents, he assigns neutral values for its identifiability and sensitivity.

John must also define the weights of causality feedback edges, or in other words, the additional impact on privacy that disclosure of an information causes on another information. In our scenario, this happens when information with high identifiability – such as name (D_6) or phone number (D_8) – is disclosed together with information with low identifiability – such as the user profile (D_1) or his pseudonym (D_2). John determines that disclosure of his name increases a lot (0.9) the identifiability of his pseudonym, moderately (0.5) increases the identifiability of his profile and greatly (0.9) increases the sensitivity of disclosing his profile. Similarly, he defines that disclosure of his telephone number largely (0.7) increases the identifiability of his pseudonym, slightly (0.3) increases the identifiability of his profile and largely increases the sensitivity of disclosing his profile. He also defines that disclosing his phone moderately (0.5) increases the identifiability of his name and vice-versa. These causality feedback edges may be neutralized in a model instance according to the task distribution of the executable workflows, as will be described in Section 6.3.3.

The weight of causality feedback edges also does not need to be manually defined by John. Based on analysis of census data or on samples from the population, organizations that defend the right to privacy can determine the potential increase in identifiability and sensitivity when different personal information is disclosed to the same entity. This correlation potential can then be used as the weights of causality feedback edges. John may import these definitions and use them as suggested or he may adapt some values to

represent his view of the privacy risks of specific correlations.

6.3.2 Service Discovery

Every service defined in Section 6.3 has a corresponding PLASTIC Service Description. Each description points to a different WSDL document that describes the service interface; the complete interfaces can be found in Appendix A. For the sake of simplicity in this scenario we assume that service descriptions are syntactic, but our middleware also supports semantic-annotated descriptions specified through SAWSDL documents as defined in [MPG+08].

As presented in Section 6.3, John does not want his friends to know that he is a member of the Cupd community. He defines that Cupd service descriptions are private both when announcing or when looking for Cupd services. In this scenario we use the EVEY Full protocol as described in Section 4.3.2, and we assume that members of the Cupd community share a secret key that allows them to privately publish services to each other. The middleware running locally in John's device automatically generates privacy-enhanced descriptions for Cupd-related services through PREMISE's *Privacy Enhancer* component.

The service announcement contains the privacy-enhanced version of the service interface, the original service interface encrypted, and a description of the Cupd encryption key. The service discovery request contain only the privacy-enhanced interface definition because requests are not encrypted. The privacy-enhanced version of the service interface in both announcements and requests is similar to the original interface, but the names of inputs and outputs are replaced by their weak hashes.

John's device does not have enough resources to act as a service directory for the local network or to act as a bridge and send messages among networks, so the middleware running locally on his device must look for an available service directory in the local network. We assume that John is able to establish his level of trust on the local service directory, by checking for instance the software version that it uses, its reputation according to other users of the network or evaluating the outcome of past requests. Service directories are connected to the bridges in the local network, so if services locally available are not sufficient to satisfy John's needs, the service directory can handle service discovery in remote networks on behalf of John.

John publishes the `MobileCupd` service description in the selected service directory using the service directory API, which enables other members of the Cupd community to run a profile match against his own profile. John also starts to look for services to realize his composition, which requires the definition of privacy preferences for service discovery requests. The middleware can automatically define these preferences based on the identifiability and sensitivity values defined for inputs required by the user-defined composition: if data has high identifiability and sensitivity the middleware can use a more cautious approach to discovery, for instance using the one-by-one propagation strategy and protecting requests with multiple layers of encryption, one for each hop traversed by the service request to the destination. On the other hand, if data required by the composition

is less sensible and identifiable, service discovery can use a more audacious approach and use a parallel propagation strategy with end-to-end encryption only. In this case, the *Discovery Preferences* component obtains identifiability and sensitivity values from the *Information Privacy Impact* component determine the privacy preferences of the service discovery request.

John can also manually define service discovery preferences such as geographical restrictions or trusted companies, and the number of results that must be obtained before terminating service discovery. Since all services use data with high sensitivity and identifiability, the middleware decides to use the one-to-one strategy for discovery propagation. John also defines that he only wants to disclose his requests to service directories located in the same city where he is, and that service discovery should proceed until three results are found. John's local middleware creates a service request containing the privacy-enhanced service description and the above discovery preferences and sends it to the selected service directory in the network.

The service directory handles John's service discovery request. Initially, the service directory tries to find a match with service announcements stored locally, and if necessary it sends the request to other directories in remote networks. Announcements are stored in the *Service Directory* and are organized as described in [MPG+08] to improve performance of semantic service matching. The *EVEY Privacy-Enhanced Matcher* implements the COCOA Conversation Integration mechanism [BGI06], that enables the middleware to recompose conversations to obtain the functionality in the user-defined composition.

The service matcher implemented in PREMISE is an extension to the PerSemSyn hybrid matcher [BRUC08], which combines EASY [MPG+08] features for semantic matching and MUSDAC [RRdLC+06] features for syntactic matching to compare fully semantic, fully syntactic and hybrid service descriptions. Our extension adds privacy awareness to the matcher, enabling it to compare privacy-enhanced versions of service descriptions. To check if two privacy-enhanced service descriptions match, the *EVEY Privacy-Enhanced Matcher* needs only to compare the codes in both requests and announcements and does not have access to the original service descriptions. In this example we use a syntactic service description, and as explained in Section 4.3.2, two privacy-enhanced syntactic descriptions are said to match if the hash of their inputs and outputs match.

John's local service directory does not find any service description matching the services required to execute the service composition. The *Controlled Request Forwarding* component uses the *Multi-radio SOAP Unicast* to send John's request to the service directory of a network one hop away. If the service discovery request used the parallel or progressive propagation strategies, the *Controlled Request Forwarding* component would use *Multi-radio SOAP Multicast* to send a message to all bridges connected to the service directory. Each bridge and service directory reports its location through OLSR HELLO messages. The *Controlled Request Forwarding* component obtains from the *Overlay Routing* component the list of service directories one hop away and their location, selects one that complies with the user-defined propagation rules, and sends John's service discovery request to that service directory only.

After sending John's request to two networks one hop away, the *Service Discovery*

component in the middleware of John's local service directory finally receives sufficient results to satisfy the user request. The service directory sends back a list with candidate service compositions based on the services available on the environment. The list contains four candidates that possibly provide the `matchProfile` operation, two that possibly provide the `invite` operation and two that possibly provide the `reservation` operation.

6.3.3 Service Selection

Service selection comprises two phases: service filtering and service choice. First PREMISE must filter out irrelevant services included in the service discovery results. Irrelevant services include false positives that the *EVEY Privacy-Enhanced Matcher* is unable to detect based only on privacy-enhanced descriptions. In the case of a syntactic service description this may happen, for instance, when the inputs, outputs and properties of two unrelated services hash to the same values, and the list of discovery results can contain one as well as the other.

Second, the user can select among the available compositions the one that causes the smallest impact over his privacy. Since the middleware can find different executable workflows that realize the user-defined abstract workflow, the user must select which of the executable workflows he wants to execute. PREMISE enables the user to consider the privacy impact of executing the workflow as a criteria for composition selection. The user does not need to manually select which workflow he wants to execute, a user profile containing user preferences (such as "always favor compositions with the smallest impact on privacy") can be used by the middleware to automatically select an executable workflow among the existing options.

For the first phase, the *Service Filtering* component of PREMISE must evaluate the list of services found by the local service directory in the local and remote networks, and must consider three possibilities. If the service description is accompanied by an encrypted file and a key description, the middleware must check if the user possesses the required key to decrypt the description. If the user does not have the key, he does not have access to the service and the description must be discarded. If the user has the key, the middleware decrypts the description and checks if the service parameters are the same as the ones the user defined. If this is not the case, the description corresponds to another service that the user can access (since he possesses the key) that coincidentally use inputs and outputs that hash to the same value as the service the user requested and must be discarded. If the unencrypted description is compatible with the original description in the user request, there is a match and the user can invoke the service. Listing 6.1 shows the pseudo-code for the first phase filtering for service selection:

```
while(results.isNotEmpty())
{
  result = results.next();
  if(result.hasKeyDescription())
  {
```

```
    if(myKeys.contain(result.getKeyDescription()))
    {
      description = decrypt(result, myKeys.getKey(result.getKeyDescription));
      if(!request.matches(result))
        results.remove(result);
    }
    else
      results.remove(result);
  }
  else
    results.remove(result);
}
```

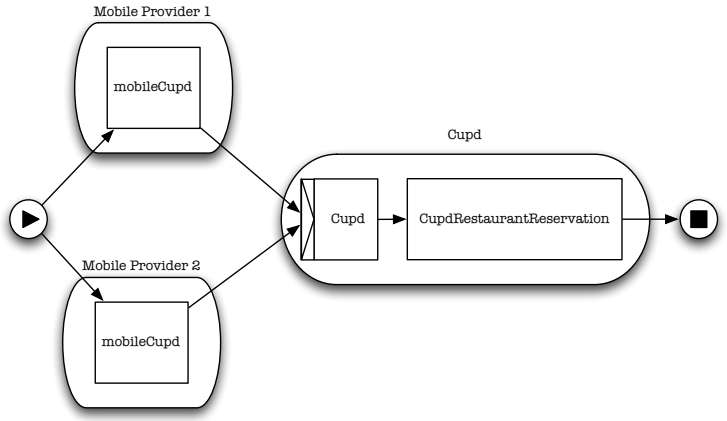
Listing 6.1: Pseudo-code for Service Selection

In our scenario, two results are filtered out for the `matchProfile` operation because John did not possess the required keys to decrypt the descriptions. This can happen for instance if those services are provided by another company that also offers a profile matching service, but to which John is not affiliated. Additionally, one result for the `invite` operation is removed because it is a false positive: another service provides an operation that coincidently requires encoded inputs that are a subset of the encoded inputs that John provides and also provides an encoded output that has the same code as the output that John requires. The original values, however, are different so the privacy-enhanced match is not verified by the original descriptions. These verifications are automatically performed by the middleware and do not require user intervention.

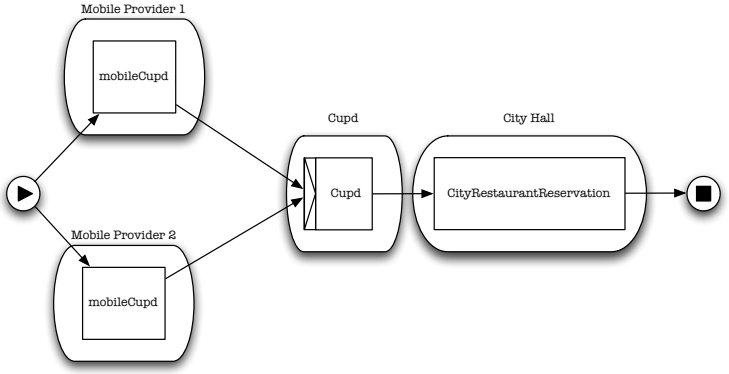
After filtering out incompatible results, the middleware keeps two instances of the `mobileCupd` service that provides a `matchProfile` operation, one instance of the `Cupd service` that provides an `invite` operation and two different services that provide a `reservation` operation: `Cupd RestaurantReservation`, a service provided by the Cupd dating community that provides reductions in restaurant reservations for meetings scheduled through Cupd, and the `CityRestaurant Reservation` service which is provided by the city where John is passing his vacations and enable tourists and residents to easily book tables in the city restaurants.

Given those services, John can choose between two executable workflows to realize his task, the one represented by Figure 6.7(a) which uses the Cupd-provided restaurant reservation service and the one represented by Figure 6.7(b) which uses the city-provided restaurant reservation service. The *Privacy-Aware Selection* component of PREMISE is responsible for evaluating the privacy impact of each composition based on the user-defined values for information privacy stored on the *Information Privacy Impact* component.

In order to compare the privacy impact of running each workflow, PREMISE creates instances of the model for evaluating the privacy impact of disclosing personal information. The model takes into account the distribution of tasks among service providers of each



(a)



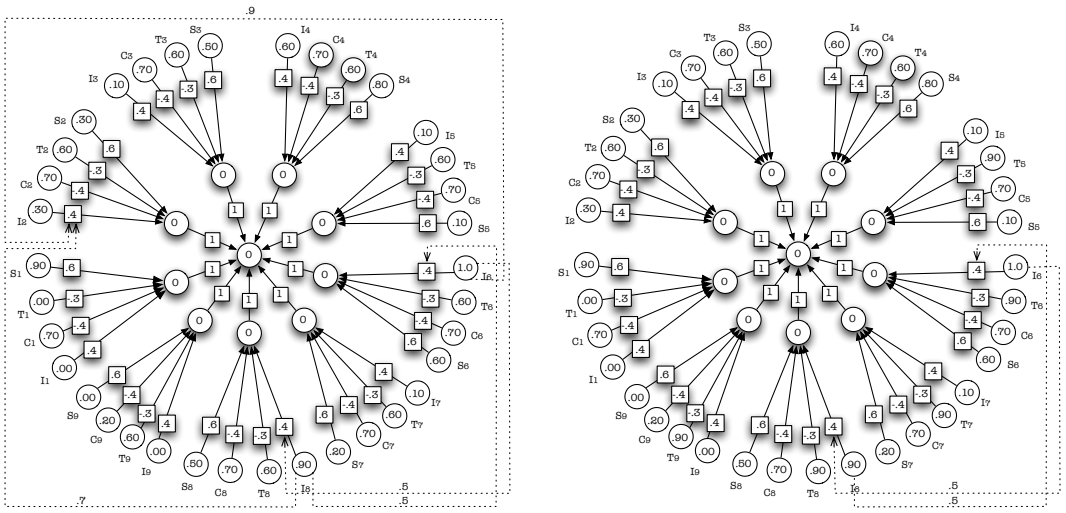
(b)

Figure 6.7: Middleware-provided Executable Workflows

workflow to give weights or neutralize causality feedback edges. The model instance also considers the trust the user has in each service provider and the convenience of disclosing each data to obtain the output required. Trust and convenience are added to the model as described in Section 5.6.2.

Both trust and convenience are defined based on past experiences. To define the trust value of a service provider, the middleware considers the past experiences between the user and the provider, and the provider reputation according to other users. Convenience can be manually defined, but may also be determined by considering how many times the same input was requested to provide the same required output in past interactions or another output related to the present user-required output. This value contributes to estimate how necessary is an input to obtain an output.

John does not trust any of the mobile providers (0.0), moderately trusts the Cupd service (0.6) and trusts a lot the City Hall (0.9). He knows that most of the data he is



(a) (b)

Figure 6.8: Modeling the Privacy Impact of Middleware-provided Executable Workflows

willing to provide is required to obtain the outputs he is looking for (0.7), except for the `meetingID` that John is not sure about the content but knows that it is not required for a restaurant reservation (0.2). Given those values, PREMISE creates the model instances shown in Figure 6.8(a) and 6.8(b).

After iterating the fuzzy cognitive maps corresponding to each executable workflow, the privacy impact of executing the first workflow is of 0.939 while execution of the second workflow results on an impact of 0.920. Both workflows have a strong impact on John's privacy which is understandable since they both require John to disclose a great amount of private information. Nonetheless, John considers that the benefits of executing the composition overcomes the harmful impact on his privacy, and decides to execute the second workflow to minimize the privacy impact of executing the service composition.

6.3.4 Service Access

John is particularly cautious about disclosing his profile. It contains information that he is usually reluctant to review to some of his closest friends, but that he thinks can help him to find the ideal partner. When accessing the `mobileCupd` service, thus, John specifies that the access should maximize privacy. He knows from past experiences that whenever he selects this option a service invocation takes longer to complete, but in this case this is not a problem: profile matching is not interactive and he does not care if he will have a response in one second or ten minutes.

Figure 6.9 shows the network topology of the configuration depicted in Figure 6.6. There are two instances of the `mobileCupd` service that John can access, and both instances are located in network N3. Users providing the `mobileCupd` service want to

remain anonymous and thus they do not possess public keys. To protect the content of service access communication, the middleware installed in John's device decides to use the privacy-enhanced service access protocol defined in Chapter 3.

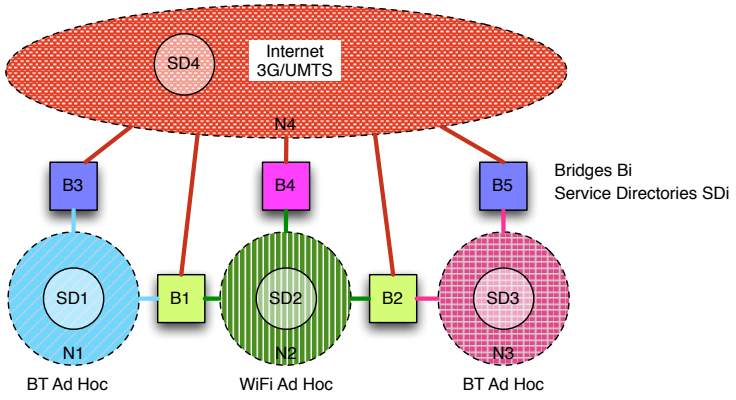


Figure 6.9: Network Topology during Service Access

The *Multi-Path Route Request* component of the PREMISE middleware installed on John's mobile device asks the local service directory for multiple paths to the network N3. Those multiple paths are simultaneously used by PREMISE to improve service access privacy by making it more resistant to eavesdropping. In the local service directory, the *Multi-path Request Handler* receives John request for multiple paths to a destination and starts a multi-path route discovery as explained in Section 3.3: the local service directory multicasts broadcasts a route request for the required destination and intermediary nodes reply with the routes they know. This process is handled by the *Multi-path Route Discovery* on bridges in cooperation with the*Multi-path DSR* component that handles overlay routing through multiple paths.

The local service directory sends back to John's device the list of available paths to the network N3 along with the network technologies and the node platforms of each node in the paths to the destination. John's local middleware thus obtains the same graph as depicted in Figure 6.9. Bridges $B3$ and $B5$ run the Windows Mobile operating system, bridges $B1$ and $B2$ run Android while bridge $B4$ runs Symbian OS. Networks $N1$ and $N3$ use Bluetooth ad hoc, $N2$ uses a WiFi ad hoc network without encryption and $N4$ uses the 3G/UMTS.

Since John defined that the `mobileCupd` service must be accessed with maximum privacy, the middleware must avoid networks that put privacy at risk. The middleware knows that WiFi networks without encryption are vulnerable to privacy attacks, so the *Multi-path Route Definition* component on John's local PREMISE middleware uses this information to find sets of paths that resist to attacks to a software platform and to WiFi ad hoc networks without encryption. In the topology depicted in Figure 6.9, only network $N2$ uses the WiFi technology without encryption, so the middleware tries to find routes that resist to attacks against network $N2$ and all nodes running the same software

platform, whichever that platform is.

The middleware's *Multi-path Route Definition* component in John's device finds a set containing three sets of routes, each one resisting to attacks to a different software platform. The set of routes R used for private communication to access the `mobileCupd` service is $R = \{\{< B3, N4, B5 >\}, \{< B1, N4, B2 >\}, \{< B1, N4, B2 >, < B1, N4, B5 >, < B3, N4, B5 >, < B3, N4, B2 >\}\}$. The first set in R resists to attacks against the software platform of nodes $B1$ and $B2$, the second resists to attacks to the software platform of nodes $B3$ and $B5$, and the third set resists to attacks to the software platform of node $B4$.

The *Split Message* component of PREMISE for regular nodes uses this set of routes to decide how the symmetric encryption key is divided and how the message is redundantly sent through multiple paths. According to the set of routes R, the key k is divided using a (3,3) secret sharing scheme resulting in shares k_1, k_2 and k_3, and all of them must be recovered by the destination to reconstruct the symmetric encryption key k.

The message M that John needs to send to the destination is encrypted with key k using the AES encryption algorithm and generates the encrypted message M_k. The middleware version of PREMISE installed in John's device knows from past interactions that the average probability of sending a message with success is 0.9, and John requires this message to be delivered with success probability 0.95. Based on the algorithm presented in Section 3.5, PREMISE determines that a (6,4) dispersion algorithm provides John's required level of reliability. The middleware splits M_k into six shares $M_1, ..., M_6$ and creates the following message shares that are sent in order to each route in R: $(k_1, M_1), (k_2, M_2), (k_3, M_3), (k_3, M_4), (k_3, M_5), (k_3, M_6)$.

Each share has on its header the full path that must be traversed to reach the destination. The *Overlay Routing* component on each intermediary bridge receives the packet, checks the packet next hop and forwards the packet to the next hop according to its list of neighbors. When the packets arrive to the destination, the *Rebuild Message* component of the destination's PREMISE checks the number of shares received and recompose the message. To send the reply message back to John's device, the node running the `mobileCupd` service executes the same algorithm on the opposite direction.

The following two tasks are provided by service providers known to John. John is a member of the Cupd community and obtained Cupd's public key when he subscribed to the service. John also has the public key of the City Hall, which he obtained in a past interaction with other services provided by the city. Data disclosed in the interactions with the following two services in John's service composition can be protected using traditional key exchange with asymmetric encryption.

6.4 Final Remarks

In this chapter we have illustrated how the various components for privacy protection introduced in this thesis can be integrated into a middleware architecture to increase user privacy while consuming and providing services in pervasive computing. The middleware architecture provides mechanisms to increase privacy during service access, discovery and

composition, and adapts to the role performed by the user's device with optional components being loaded only when needed, to reduce the middleware overhead.

PREMISE for regular nodes contain only the components required to announce and discover services in a pervasive computing environment. Those components are not continuously running or exchanging messages, and thus do not impose a fixed overhead on the user's device. PREMISE for bridges and service directories contain additional components that are required to create the infrastructure that facilitates service provision and consumption in highly mobile and dynamic networks. These middleware components must be executed only by a small number of more powerful nodes that possess additional resources and want to share them with the network.

The middleware architecture provides features such as privacy-enhanced service discovery and privacy-aware service composition, that help to reduce the privacy impact caused by the middleware itself and enable the user to better control information disclosure while using the middleware. It also provides a privacy-enhanced service access feature that offers an alternative mechanism to increase the confidentiality of messages during service access when public key encryption is undesirable or unavailable. PREMISE, the middleware presented here, provides a solid base for a service-oriented middleware with privacy protection mechanisms for pervasive computing, and can be extended with complementary features to enable users and service providers to control with even more details how they disclose personal data when providing and consuming pervasive services.

No phone

No phone

I just want to be alone today.

Cake, "No Phone"

7

Final Remarks and Perspectives

The right to privacy is fundamental in human societies. Privacy is essential for building and managing social relationships, and is necessary for learning and for intellectual growth. We are living the beginning of a new society, where information becomes the most important asset and society is organized so as to facilitate its flow. It is crucial that the cultural revolution leading to the creation of this Knowledge Society preserves the basic human rights, and privacy is one of them.

Since their invention and throughout their evolution, computers posed a threat to privacy. As information in general becomes easier to organize, to transfer, to store and to collect, the same happens to personal information and data that once was lost as a normal result of regular social interactions is now stored and searchable. Individuals who have the habit of interacting with humans have problems to manage their privacy when interacting with computers because computers extend human capacities and individuals cannot predict the consequences on their private lives when computers handle personal data.

As we evolve towards this Knowledge Society, computers become more and more present in our daily activities. One example of this dissemination is the pervasive computing view of computers widely deployed, naturally helping users to realize their everyday tasks and simplifying and augmenting the possibilities for interaction with other users

and with the surrounding environment. Since a great number of user activities in pervasive computing involves computers, a large amount of personal data can be digitalized, stored and analyzed when individuals use pervasive application. The problem of managing privacy when interacting with computers is even more critical in pervasive computing.

The harmful consequences of being unable to control private data disclosure are real. A crime that is becoming increasingly popular is identity theft. An identity theft happens when a thief obtains so much personal data about a person that he becomes able to impersonate that person. After impersonating someone else, the thief can more easily perform bank frauds or other crimes that require some sort of authentication. This problem does not affect only persons who lack computer knowledge and could be more easily deceived into incorrectly disclosing personal data. Surprisingly, even prominent scientists in the area of identity management can be victims of an identity theft [Cam08]. Everybody is at risk and anyone can be a target.

Fortunately, companies and the general public are becoming aware of the risks of revealing personal data without proper control and are beginning to recognize the value of privacy. One example was the recent acquisition of Credentica by Microsoft [Fon08]. Credentica was a company founded by Stefan Brands to develop solutions for privacy-aware digital certificates, based on the work of his thesis [Bra00]. This move by a major software development company is a sign that there is an interest from the mainstream software consumers in privacy protection technologies.

7.1 Contribution

In this thesis we contribute with the effort of protecting privacy in the future Knowledge Society by proposing to build privacy controls in the software infrastructure of pervasive computing systems. As SOA is an attractive approach to build pervasive applications, we designed a service-oriented middleware for pervasive computing containing components and protocols that increase privacy and give users more control over private information disclosure when providing and consuming services in pervasive computing.

After analyzing the typical software architecture of pervasive computing environments we concluded that privacy protection at the middleware layer is a compelling method to increase user privacy in pervasive computing. Since the middleware runs on top of heterogeneous platforms and acts as the interface between applications and the underlying software layers, the middleware can avoid that privacy attacks directed to lower layers reach applications. The middleware can also provide tools and mechanisms to applications that facilitate the development of privacy respectful services. However, the very middleware features that enable provision and consumption of services in pervasive computing introduce risks to privacy that affect any application using the middleware.

In order to identify the shortcomings of existing privacy protection technologies when applied to service-oriented pervasive computing, we reviewed privacy research results in different areas of computer science and we consider that the requirements of service-oriented pervasive computing creates new challenges for privacy protection. Existing results fall

short to address issues such as mobility or lack of an infrastructure, or fail to mitigate the privacy risks introduced by service-oriented features. Our contribution complements existing research in privacy by introducing a middleware that contains mechanisms and protocols to increase privacy protection during service access, service discovery and service composition in pervasive computing.

One of the characteristics of pervasive computing environments is that they are open and cannot rely on an infrastructure. Additionally, users provide services to each other and cooperate to perform some tasks. Traditional privacy-enhancement solutions for service access are based on public key encryption, and assume that an infrastructure for certificate generation and validation is available. They also assume that service providers must be authenticated by clients. In pervasive computing this is not always the case, and thus alternative methods for privacy-enhanced service access are necessary. The middleware proposed in this thesis supports a protocol for increasing data confidentiality during service access that explores the potentially multiple paths between a client and a service provider to privately send a message. The protocol resists to attackers controlling all nodes running one of the software platforms available and also controlling a given network technology, allowing clients and service providers to communicate in private whenever public key encryption is undesirable.

Service discovery in pervasive computing also poses a threat to user privacy. Service announcements and requests contain personal information that can be used to infer personal details such as user activities and preferences. Service directories must be highly available and are traditionally widely distributed in untrusted user devices, which is critical for privacy since they handle service announcements and requests. We thus introduce a privacy-enhanced service discovery protocol that reduces sensitivity of service descriptions and enables deployment of service directories in untrusted nodes. The protocol encodes service descriptions in such a way that multiple original descriptions can generate the same privacy-enhanced description. Service directories and other entities that have access only to privacy-enhanced service descriptions are unable to identify the original description. The protocol also contains mechanisms to reduce exposure of service discovery requests to untrusted entities when performing discovery in remote networks.

Finally, mechanisms for service composition in pervasive computing must be flexible to cope with the environment's dynamics and openness, but this flexibility introduces additional risks to privacy. In pervasive computing, users join and leave the system frequently and discover resources and services available at run time. In this setting it is very unlikely that clients are able to discover the exact user-defined service composition, and flexible composition mechanisms exist that use available services to obtain the same functionality as defined by the user but using a possibly different workflow. This approach increases the power of service composition but also modifies the partition of data among service providers, which may cause unexpected effects on user privacy. We provide a model for reasoning about the privacy impact of disclosing personal information. The models representing each information can be combined to determine the privacy impact of executing a service composition, enabling the user to decide if the privacy impact of executing the composition is worth the benefits and which composition to choose among

different compositions that are functionally equivalent but that use different workflows.

We believe that the middleware architecture proposed in this thesis offers a solid base for privacy respectful service-oriented pervasive computing. The components for privacy protection integrated into the architecture reduce the privacy risks caused by basic functionalities for service provision and consumption in pervasive computing, enabling users and applications to better control disclosure of personal data. As the middleware reduces the risks for privacy invasion when offering and providing services, applications can focus on privacy protection against application-specific issues, which facilitates development of pervasive computing applications that respect the privacy of users.

7.2 Publications

The subject studied in this thesis is the focus of a number of papers published in workshops and international conferences. The following list contains all published articles describing results from our research.

- Roberto Speicys Cardoso, Valérie Issarny. *Architecting Pervasive Computing Systems for Privacy : A Survey.* In Proceedings of 6th Working IEEE/IFIP Conference on Software Architecture (WICSA). Mumbai, India, January 2007.

  This article discusses privacy protection techniques for service-oriented pervasive computing and proposes the categorization of privacy-related research used in Chapter 2.

- Roberto Speicys Cardoso and Mauro Caporuscio. *Exploring Multi-path Communication in Hybrid Wireless Ad Hoc Networks.* In 1st International Workshop on Ad-hoc Ambient Computing, Sophia Antipolis, France, September 2008.

  This article discusses some of the issues and ideas for increasing privacy protection during service access using multi-path routing. The ideas discussed in this article led to the protocol presented in Chapter 3.

- Roberto Speicys Cardoso, Pierre-Guillaume Raverdy, Valérie Issarny. *A Privacy-Aware Service Discovery Middleware for Pervasive Environments.* In IFIPTM 2007 Joint iTrust and PST Conferences on Privacy, Trust Management and Security, Moncton, Canada, August 2007 – **Best Student Paper Award**.

  The paper discusses how to improve privacy in distributed service discovery protocols, and discusses the privacy issues that appear when service discovery requests are sent to remote untrusted networks. This article describes the basic privacy protection mechanisms used by the Distributed EVEY protocol presented in Chapter 4.

- Sonia Ben-Mokhtar, Pierre-Guillaume Raverdy, Roberto Speicys Cardoso, Aitor Urbieta and Nikolaos Georgantas. *Discovering Social Services in Pervasive Environments with Privacy – Demonstration Paper.* In ACM/IFIP/USENIX 8th International Middleware Conference (Middleware 2007), Newport Beach, USA, November 2007.

This demonstration shows a service discovery scenario that uses syntactic and semantic service descriptions, without privacy protection and privacy-enhanced. The goal is to demonstrate the benefits of semantic service discovery and the protection provided by privacy-enhanced service descriptions. It uses the protocol presented in Chapter 4 and some of the ideas in this demonstration served as a base for the scenario in Chapter 6.

- Roberto Speicys Cardoso, Sonia Ben-Mokhtar, Aitor Urbieta and Valérie Issarny. *EVEY: Enhancing Privacy of Service Discovery in Pervasive Computing.* In ACM/IFIP/USENIX 8th International Middleware Conference (Middleware 2007), Newport Beach, USA, November 2007.

  The article presents the initial design of the EVEY Full service discovery protocol, described in more detail in this thesis in Chapter 4.

- Sonia Ben Mokhtar, Pierre-Guillaume Raverdy, Aitor Urbieta and Roberto Speicys Cardoso. *Interoperable Semantic & Syntactic Service Matching for Ambient Computing Environments.* In 1st International Workshop on Ad-hoc Ambient Computing, Sophia Antipolis, France, September 2008.

  This paper mainly discusses a hybrid service discovery protocol for pervasive computing. We contributed to the protocol definition as part of the research on privacy-enhanced semantic and syntactic service discovery, which led to creation of the EVEY protocol presented in Chapter 4.

- Roberto Speicys Cardoso, Valérie Issarny. *A Model for Reasoning About the Privacy Impact of Composite Service Execution in Pervasive Computing.* In IFIPTM 2008 Joint iTrust and PST Conferences on Privacy, Trust Management and Security, Trondheim, Norway, June 2008.

  This article presents the model for reasoning about the privacy impact of a service composition, which is discussed to a greater extent in Chapter 5 of this document.

This thesis was also the opportunity of collaborating with other researchers on subjects relevant to our research but not strictly related to the theme of this thesis. The following publication is the result of one such collaboration.

- Natallia Kokash, Roberto Speicys Cardoso, Pierre-Guillaume Raverdy and Valérie Issarny. *A Flexible QoS-Aware Routing Protocol for Infrastructure-less B3G Networks.* In 24th Annual ACM Symposium on Applied Computing - Mobile Computing and Applications Track, Hawaii, USA, March 2009.

  This article proposes a multi-radio routing protocol for MANETs that efficiently proactively keeps the routes with optimal throughput, delay and cost. This protocol was created as part of our effort to develop an energy-efficient multi-radio routing protocol.

7.3 Perspectives

The middleware architecture presented in this thesis is a first step on introducing privacy protection mechanisms into the pervasive computing software architecture, but a lot remains to be done to build a software layer for pervasive computing that prevents the creation of privacy invasive applications. In this section we discuss some short-term and long-term perspectives for improving privacy in pervasive computing systems.

The privacy-enhanced service access protocol proposed in Chapter 3 is greatly dependent on bridge availability and heterogeneity. If the pervasive network contains few bridges or if all bridges run the same platform, the middleware cannot find enough heterogeneous paths and an attacker that compromises one type of software platform can invade user privacy. To remediate this situation, users must have a strong incentive to volunteer as bridges, and elected bridges must be as heterogeneous as possible. On the other hand, the number of bridges must be restricted to a fraction of all nodes since bridges are responsible for routing and a high number of bridges can cause an unnecessary communication overhead.

The middleware, thus, must provide an incentive mechanism that rewards users volunteering as bridges. Many economic-based incentive mechanisms exist in the literature for applications such as peer-to-peer file sharing [Coh03a] and service allocation in pervasive computing [LI04a]. These mechanisms aim to maximize the system's efficiency by giving incentives that reward entities behaving in a way that is beneficial to the whole system. The incentives are usually related to the system's purpose, for instance rewarding nodes that upload often or service providers that correctly announce their quality of service in the examples above.

Adapting incentive mechanisms to bridge election raises a number of issues. First we must define which incentives give to users volunteering as bridges. Since this mechanism is implemented at the middleware layer, users can have incentives such as priority in traffic, more service discovery results or a higher number of alternative routes to a destination. Efficient and distributed implementation of these incentives in HMANETs is an issue.

Besides that, the incentives must be adapted according to the local and the overall network topology. When a user volunteers to act as a bridge, the impact on the network caused by bridge deployment depends on variables such as the number of bridges already running in the local network, the operating system they use and the additional networks that the bridge connects. The incentives must be distributed according to the positive impact caused by the bridge deployment, and how to effectively achieve this is also an issue.

We must also investigate mechanisms for data usage control in pervasive computing. Although avoiding specific uses of personal data seems unfeasible given current mobile devices and the openness of pervasive environments such as HMANETs, we can considerably improve user knowledge on how and when service providers use his personal data. Given the difficulty of ensuring that personal data is not going to be abused, a mechanism that notifies the user whenever his personal data is accessed can help with service provider accountability in case of privacy invasions.

We plan to better investigate how mobile code can be used to improve user awareness on how and when his personal data is used. Service providers can send mobile code to user devices to avoid the need of private information disclosure by locally executing methods that require personal data and give a result that is less sensitive than the information required to execute it. Mobile code can also be used by clients to encapsulate the data they send to service providers and be informed whenever that data is accessed. Securely running mobile code on the service provider side may require service provider cooperation and is dependent on how much the client trusts a service provider, so mobile code protection must take these issues into account.

In this thesis we studied the particular case of a service-oriented pervasive computing environment, but privacy protection is necessary for any pervasive computing architecture, either using the SOA architectural style or not. We plan to create architecture-agnostic privacy protection mechanisms designed to work in pervasive computing systems regardless of their architectural style. Those mechanisms would increase privacy in typical pervasive computing features such as context provision, notification or mobility management. They would have to handle the heterogeneity of those features to offer an homogeneous interface for applications. This privacy protection layer could even offer additional architectural-agnostic privacy related functionalities such as image filters and location modifiers. These middleware-provided functionalities would allow different applications to use the same mechanisms for privacy protection, avoiding duplication of code.

Finally, an important obstacle for privacy protection in computer systems in general and in pervasive computing in particular is how to design a natural interface that enables users to easily manage their privacy. As pointed out by [Ack00], whenever social processes must be modeled by an computing system there is a *social-technical gap*, a difference between the social requirements the system must support and those that it can support technically. Privacy management is essentially a social activity, so it also suffers from this social-technical gap. If users must explicitly define what they instinctively do to manage their privacy, the information space becomes so big that it overwhelms the user with options. The interface problem for privacy protection is not only a graphical interface problem but it is much more complex than that, and requires systems that learns with the user activities and adapts to new situations without user intervention.

A

Scenario Service Descriptions

A.1 MobileCupd WSDL Description

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
  xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
  xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
  xmlns:ns1="http://org.apache.axis2/xsd"
  xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
  xmlns:ax21="http://cupd/xsd" xmlns:ns="http://cupd"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  targetNamespace="http://cupd">
    <wsdl:documentation>
    Find a partner while moving with MobileCupd!
```

```
  </wsdl:documentation>
    <wsdl:types>
        <xs:schema xmlns:ax22="http://cupd/xsd"
        attributeFormDefault="qualified" elementFormDefault="qualified"
        targetNamespace="http://cupd">
    <xs:import namespace="http://cupd/xsd"/>
    <xs:element name="matchProfile">
        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="0" name="param0" nillable="true"
                type="ax22:CupdProfile"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="matchProfileResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="0" name="return" nillable="true"
                type="ax22:CupdProfile"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
        <xs:schema attributeFormDefault="qualified"
        elementFormDefault="qualified"
        targetNamespace="http://cupd/xsd">
    <xs:complexType name="CupdProfile">
        <xs:sequence/>
    </xs:complexType>
</xs:schema>
    </wsdl:types>
    <wsdl:message name="matchProfileRequest">
        <wsdl:part name="parameters" element="ns:matchProfile"/>
    </wsdl:message>
    <wsdl:message name="matchProfileResponse">
        <wsdl:part name="parameters"
```

```
        element="ns:matchProfileResponse"/>
    </wsdl:message>
    <wsdl:portType name="MobileCupdPortType">
        <wsdl:operation name="matchProfile">
            <wsdl:input message="ns:matchProfileRequest"
            wsaw:Action="urn:matchProfile"/>
            <wsdl:output message="ns:matchProfileResponse"
            wsaw:Action="urn:matchProfileResponse"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="MobileCupdSoap11Binding"
    type="ns:MobileCupdPortType">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
        style="document"/>
        <wsdl:operation name="matchProfile">
            <soap:operation soapAction="urn:matchProfile"
            style="document"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:binding name="MobileCupdSoap12Binding"
    type="ns:MobileCupdPortType">
        <soap12:binding
        transport="http://schemas.xmlsoap.org/soap/http"
        style="document"/>
        <wsdl:operation name="matchProfile">
            <soap12:operation soapAction="urn:matchProfile"
            style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
```

```
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:binding name="MobileCupdHttpBinding"
    type="ns:MobileCupdPortType">
        <http:binding verb="POST"/>
        <wsdl:operation name="matchProfile">
            <http:operation location="MobileCupd/matchProfile"/>
            <wsdl:input>
                <mime:content type="text/xml" part="matchProfile"/>
            </wsdl:input>
            <wsdl:output>
                <mime:content type="text/xml" part="matchProfile"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="MobileCupd">
        <wsdl:port name="MobileCupdHttpSoap11Endpoint"
        binding="ns:MobileCupdSoap11Binding">
            <soap:address
            location="http://192.168.206.1:8080/ScenarioWeb/
            services/MobileCupd.MobileCupdHttpSoap11Endpoint/"/>
        </wsdl:port>
        <wsdl:port name="MobileCupdHttpSoap12Endpoint"
        binding="ns:MobileCupdSoap12Binding">
            <soap12:address
            location="http://192.168.206.1:8080/ScenarioWeb/
            services/MobileCupd.MobileCupdHttpSoap12Endpoint/"/>
        </wsdl:port>
        <wsdl:port name="MobileCupdHttpEndpoint"
        binding="ns:MobileCupdHttpBinding">
            <http:address
            location="http://192.168.206.1:8080/ScenarioWeb/
            services/MobileCupd.MobileCupdHttpEndpoint/"/>
```

```
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

A.2 Cupd WSDL Description

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:ns1="http://org.apache.axis2/xsd"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:ns="http://cupd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 targetNamespace="http://cupd">
    <wsdl:documentation>
    Send an invitation to your date in Cupd
  </wsdl:documentation>
    <wsdl:types>
        <xs:schema attributeFormDefault="qualified"
        elementFormDefault="qualified"
        targetNamespace="http://cupd">
    <xs:element name="invite">
        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="0" name="param0" type="xs:int"/>
                <xs:element minOccurs="0" name="param1"
                 nillable="true" type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="inviteResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="0" name="return" type="xs:int"/>
```

```
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
    </wsdl:types>
    <wsdl:message name="inviteRequest">
        <wsdl:part name="parameters" element="ns:invite"/>
    </wsdl:message>
    <wsdl:message name="inviteResponse">
        <wsdl:part name="parameters" element="ns:inviteResponse"/>
    </wsdl:message>
    <wsdl:portType name="CupdPortType">
        <wsdl:operation name="invite">
            <wsdl:input message="ns:inviteRequest"
            wsaw:Action="urn:invite"/>
            <wsdl:output message="ns:inviteResponse"
            wsaw:Action="urn:inviteResponse"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="CupdSoap11Binding" type="ns:CupdPortType">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
        style="document"/>
        <wsdl:operation name="invite">
            <soap:operation soapAction="urn:invite" style="document"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:binding name="CupdSoap12Binding"
    type="ns:CupdPortType">
        <soap12:binding
        transport="http://schemas.xmlsoap.org/soap/http"
```

```
        style="document"/>
        <wsdl:operation name="invite">
            <soap12:operation soapAction="urn:invite"
            style="document"/>
            <wsdl:input>
                <soap12:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap12:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:binding name="CupdHttpBinding" type="ns:CupdPortType">
        <http:binding verb="POST"/>
        <wsdl:operation name="invite">
            <http:operation location="Cupd/invite"/>
            <wsdl:input>
                <mime:content type="text/xml" part="invite"/>
            </wsdl:input>
            <wsdl:output>
                <mime:content type="text/xml" part="invite"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="Cupd">
        <wsdl:port name="CupdHttpSoap11Endpoint"
        binding="ns:CupdSoap11Binding">
            <soap:address location="http://192.168.206.1:8080/
            ScenarioWeb/services/Cupd.CupdHttpSoap11Endpoint/"/>
        </wsdl:port>
        <wsdl:port name="CupdHttpSoap12Endpoint"
        binding="ns:CupdSoap12Binding">
            <soap12:address location="http://192.168.206.1:8080/
            ScenarioWeb/services/Cupd.CupdHttpSoap12Endpoint/"/>
        </wsdl:port>
        <wsdl:port name="CupdHttpEndpoint"
```

```
        binding="ns:CupdHttpBinding">
            <http:address location="http://192.168.206.1:8080/
            ScenarioWeb/services/Cupd.CupdHttpEndpoint/"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

A.3 City Hall Restaurant Reservation WSDL Description

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:ns1="http://org.apache.axis2/xsd"
xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
xmlns:ns="http://cupd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
targetNamespace="http://cupd">
    <wsdl:documentation>
    This is the restaurant reservation service provided by the City of X
  </wsdl:documentation>
    <wsdl:types>
        <xs:schema attributeFormDefault="qualified"
        elementFormDefault="qualified" targetNamespace="http://cupd">
    <xs:element name="reservation">
        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="0" name="param0" nillable="true"
                 type="xs:dateTime"/>
                <xs:element minOccurs="0" name="param1" nillable="true"
                type="xs:string"/>
                <xs:element minOccurs="0" name="param2" type="xs:int"/>
                <xs:element minOccurs="0" name="param3" nillable="true"
                type="xs:string"/>
            </xs:sequence>
```

```
        </xs:complexType>
    </xs:element>
    <xs:element name="reservationResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="0" name="return" type="xs:int"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
    </wsdl:types>
    <wsdl:message name="reservationRequest">
        <wsdl:part name="parameters" element="ns:reservation"/>
    </wsdl:message>
    <wsdl:message name="reservationResponse">
        <wsdl:part name="parameters" element="ns:reservationResponse"/>
    </wsdl:message>
    <wsdl:portType name="CityRestaurantReservationPortType">
        <wsdl:operation name="reservation">
            <wsdl:input message="ns:reservationRequest"
            wsaw:Action="urn:reservation"/>
            <wsdl:output message="ns:reservationResponse"
            wsaw:Action="urn:reservationResponse"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="CityRestaurantReservationSoap11Binding"
    type="ns:CityRestaurantReservationPortType">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
         style="document"/>
        <wsdl:operation name="reservation">
            <soap:operation soapAction="urn:reservation"
            style="document"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
```

```
            <soap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="CityRestaurantReservationSoap12Binding"
type="ns:CityRestaurantReservationPortType">
    <soap12:binding
    transport="http://schemas.xmlsoap.org/soap/http"
    style="document"/>
    <wsdl:operation name="reservation">
        <soap12:operation soapAction="urn:reservation"
        style="document"/>
        <wsdl:input>
            <soap12:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap12:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="CityRestaurantReservationHttpBinding"
type="ns:CityRestaurantReservationPortType">
    <http:binding verb="POST"/>
    <wsdl:operation name="reservation">
        <http:operation
        location="CityRestaurantReservation/reservation"/>
        <wsdl:input>
            <mime:content type="text/xml" part="reservation"/>
        </wsdl:input>
        <wsdl:output>
            <mime:content type="text/xml" part="reservation"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:service name="CityRestaurantReservation">
    <wsdl:port name="CityRestaurantReservationHttpSoap11Endpoint"
```

```
        binding="ns:CityRestaurantReservationSoap11Binding">
            <soap:address location="http://192.168.206.1:8080/
            ScenarioWeb/services/CityRestaurantReservation.
            CityRestaurantReservationHttpSoap11Endpoint/"/>
        </wsdl:port>
        <wsdl:port name="CityRestaurantReservationHttpSoap12Endpoint"
        binding="ns:CityRestaurantReservationSoap12Binding">
            <soap12:address location="http://192.168.206.1:8080/
            ScenarioWeb/services/CityRestaurantReservation.
            CityRestaurantReservationHttpSoap12Endpoint/"/>
        </wsdl:port>
        <wsdl:port name="CityRestaurantReservationHttpEndpoint"
        binding="ns:CityRestaurantReservationHttpBinding">
            <http:address location="http://192.168.206.1:8080/
            ScenarioWeb/services/CityRestaurantReservation.
            CityRestaurantReservationHttpEndpoint/"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

A.4 Cupd Restaurant Reservation WSDL Description

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
 xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
 xmlns:ns1="http://org.apache.axis2/xsd"
 xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl"
 xmlns:ns="http://cupd" xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
 targetNamespace="http://cupd">
    <wsdl:documentation>
    Get a rebate reserving your restaurant through Cupd!
  </wsdl:documentation>
    <wsdl:types>
```

```
        <xs:schema attributeFormDefault="qualified"
        elementFormDefault="qualified" targetNamespace="http://cupd">
    <xs:element name="reservation">
        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="0" name="param0" nillable="true"
                type="xs:dateTime"/>
                <xs:element minOccurs="0" name="param1" nillable="true"
                type="xs:string"/>
                <xs:element minOccurs="0" name="param2" type="xs:int"/>
                <xs:element minOccurs="0" name="param3" nillable="true"
                type="xs:string"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="reservationResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element minOccurs="0" name="return" type="xs:int"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
</xs:schema>
    </wsdl:types>
    <wsdl:message name="reservationRequest">
        <wsdl:part name="parameters" element="ns:reservation"/>
    </wsdl:message>
    <wsdl:message name="reservationResponse">
        <wsdl:part name="parameters" element="ns:reservationResponse"/>
    </wsdl:message>
    <wsdl:portType name="CupdRestaurantReservationPortType">
        <wsdl:operation name="reservation">
            <wsdl:input message="ns:reservationRequest"
            wsaw:Action="urn:reservation"/>
            <wsdl:output message="ns:reservationResponse"
            wsaw:Action="urn:reservationResponse"/>
```

```
    </wsdl:operation>
</wsdl:portType>
<wsdl:binding name="CupdRestaurantReservationSoap11Binding"
type="ns:CupdRestaurantReservationPortType">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
    style="document"/>
    <wsdl:operation name="reservation">
        <soap:operation soapAction="urn:reservation"
        style="document"/>
        <wsdl:input>
            <soap:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="CupdRestaurantReservationSoap12Binding"
type="ns:CupdRestaurantReservationPortType">
    <soap12:binding
    transport="http://schemas.xmlsoap.org/soap/http"
    style="document"/>
    <wsdl:operation name="reservation">
        <soap12:operation soapAction="urn:reservation"
        style="document"/>
        <wsdl:input>
            <soap12:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap12:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
<wsdl:binding name="CupdRestaurantReservationHttpBinding"
type="ns:CupdRestaurantReservationPortType">
    <http:binding verb="POST"/>
```

```
        <wsdl:operation name="reservation">
            <http:operation
            location="CupdRestaurantReservation/reservation"/>
            <wsdl:input>
                <mime:content type="text/xml" part="reservation"/>
            </wsdl:input>
            <wsdl:output>
                <mime:content type="text/xml" part="reservation"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="CupdRestaurantReservation">
        <wsdl:port name="CupdRestaurantReservationHttpSoap11Endpoint"
        binding="ns:CupdRestaurantReservationSoap11Binding">
            <soap:address location="http://192.168.206.1:8080/
            ScenarioWeb/services/CupdRestaurantReservation.
            CupdRestaurantReservationHttpSoap11Endpoint/"/>
        </wsdl:port>
        <wsdl:port name="CupdRestaurantReservationHttpSoap12Endpoint"
        binding="ns:CupdRestaurantReservationSoap12Binding">
            <soap12:address location="http://192.168.206.1:8080/
            ScenarioWeb/services/CupdRestaurantReservation.
            CupdRestaurantReservationHttpSoap12Endpoint/"/>
        </wsdl:port>
        <wsdl:port name="CupdRestaurantReservationHttpEndpoint"
        binding="ns:CupdRestaurantReservationHttpBinding">
            <http:address location="http://192.168.206.1:8080/
            ScenarioWeb/services/CupdRestaurantReservation.
            CupdRestaurantReservationHttpEndpoint/"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Bibliography

[ABF+04] Rakesh Agrawal, Roberto Bayardo, Christos Faloutsos, Jerry Kiernan, Ralf Rantzau, and Ramakrishnan Srikant. Auditing Compliance with a Hippocratic Database. In *VLDB '04: Proceedings of the 13th International Conference on Very Large Data Bases*, pages 516–527, 2004.

[ABJ89] R. Agrawal, A. Borgida, and H. V. Jagadish. Efficient Management of Transitive Relationships in Large Data and Knowledge Bases. In *ACM SIGMOD International Conference on Management of Data (SIGMOD '89)*, 1989.

[Ack00] Mark S. Ackerman. The Intellectual Challenge of CSCW: the Gap between Social Requirements and Technical Feasibility. *Human-Computer Interaction*, 15(2), 2000.

[ACLGM93] E. Ayanoglu, I. Chih-Lin, R. D. Gitlin, and J. E. Mazo. Diversity Coding for Transparent Self-Healing and Fault-Tolerant Communication Networks. *IEEE Transactions on Communications*, 41(11), 1993.

[Age08a] Agence France-Presse (AFP). Brazil Senate Orders Google to Identify Website Pedophiles, April 2008. Available at `afp.google.com/article/ALeqM5gU98JC3pwFFdUMNRnqQ_CNA7u25g`.

[Age08b] Agence France-Presse (AFP). Google Hands Over Data on Suspected Pedophiles to Brazil, April 2008. Available at `afp.google.com/article/ALeqM5iWDAYEltMmXaowKfySppN_br-Acg`.

[Agu02] Jose Aguilar. Adaptive Random Fuzzy Cognitive Maps. In *IBERAMIA 2002: Proceedings of the 8th Ibero-American Conference on AI*, 2002.

[AHKB03] W. M. P. Van Der Aalst, A. H. M. Ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(1), 2003.

[AKBLN89] H. Aït-Kaci, R. Boyer, P. Lincoln, and R. Nasr. Efficient Implementation of Lattice Operations. *ACM Transactions on Programming Languages and Systems*, 11(1), 1989.

[AKSX02] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic Databases. In *28th International Conference on Very Large Databases (VLDB)*, 2002.

[All99] Anita L. Allen. Coercing Privacy. *William and Mary Law Review*, 40, 1999.

[AM00] Gregory D. Abowd and Elizabeth D. Mynatt. Charting Past, Present, and Future Research in Ubiquitous Computing. *ACM Transactions on Computer-Human Interaction*, 7(1), March 2000.

[AMCK+02] J. Al-Muhtadi, R. Campbell, A. Kapadia, M. Dennis Mickunas, and S. Yi. Routing Through the Mist: Privacy Preserving Communication in Ubiquitous Computing Environments. In *ICDCS'02 - 22nd IEEE International Conference*, pages 74–83, 2002.

[AMM07] E. Al-Masri and Q. H. Mahmoud. QoS-based Discovery and Ranking of Web Services. In *ICCN'07: IEEE 16th International Conference on Computer Communications and Networks*, 2007.

[AW89] Nabil R. Adam and John C. Worthmann. Security-control Methods for Statistical Databases: a Comparative Study. *ACM Computing Surveys*, 21(4):515–556, 1989.

[AWW05] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. Wireless Mesh Networks: A Survey. *Computer Networks and ISDN Systems*, 47(4), 2005.

[Axe76] R. Axelrod. *Structure of Decision: The Cognitive Maps of Political Elites*. Princeton University Press, 1976.

[B+05a] J. Beatty et al. *Web Services Dynamic Discovery (WS-Discovery)*, April 2005. Available at `specs.xmlsoap.org/ws/2005/04/discovery/ws-discovery.pdf`.

[B+05b] Jérôme Bindé et al. Towards Knowledge Societies. UNESCO World Report, UNESCO, 2005.

[BBC07] BBC. Monster attack steals user data, August 2007. Available at `news.bbc.co.uk/2/hi/technology/6956349.stm`.

[BE01] R. A. Botha and J. H. P. Eloff. Separation of Duties for Access Control Enforcement in Workflow Environments. *IBM Systems Journal*, 40(3), 2001.

[Bec80] Leland L. Beck. A Security Mechanism for Statistical Databases. *ACM Transaction on Database Systems*, 5(3):316–3338, 1980.

[BEG00] M. Boyle, C. Edwards, and S. Greenberg. The Effects of Filtered Video on Awareness and Privacy. In *Proceedings of ACM CSCW '00*, pages 1–10, 2000.

[Ben95] Jeremy Bentham. *The Panopticon Writings*. Verso, 1995.

[Ben07] Sonia Ben-Mokhtar. *Semantic Middleware for Service-Oriented Pervasive Computing*. PhD thesis, Université de Paris VI, December 2007.

[BFK01] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: a System for Anonymous and Unobservable Internet Access. In *International workshop on Designing privacy enhancing technologies*, January 2001.

[BGI06] S. Ben Mokhtar, N. Georgantas, and V. Issarny. COCOA: COnversation-based Service COmposition in PervAsive Computing Environments. In *ICPS'06: Proceedings of the IEEE International Conference on Pervasive Services*, 2006.

[BJ06] M. Barbaro and T. Zeller Jr. A Face Is Exposed for AOL Searcher No. 4417749. *The New York Times*, August 2006.

[BK02] Abraham Bernstein and Mark Klein. Towards High-Precision Service Retrieval. In *ISWC '02: Proceedings of the First International Semantic Web Conference on The Semantic Web*, 2002.

[BKGI06] S. Ben Mokhtar, A. Kaul, N. Georgantas, and V. Issarny. Efficient Semantic Service Discovery in Pervasive Computing Environments. In *Proceedings of Middleware'06*, 2006.

[BLHL01] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.

[Blo70] B. H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Communication of the ACM*, 13(7), 1970.

[BMAP03] H. Badis, A. Munaretto, K. A. Agha, and G. Pujolle. QoS for Ad hoc Networking Based on Multiple Metrics: Bandwidth and Delay. In *Proceedings of the Fifth IFIP-TC6 International Conference on Mobile and Wireless Communications Networks*, 2003.

[BN89] D. F. C. Brewer and M. J. Nash. The Chinese Wall Security Policy. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1989.

[BP05] Antonio Brogi and Razvan Popescu. Towards Semi-automated Workflow-Based Aggregation of Web Services. In *ICSOC 2005: Proceedings of the Third International Conference on Service-Oriented Computing*, 2005.

[BP06] Antonio Brogi and Razvan Popescu. From BPEL Processes to YAWL Workflows. In *Proceedings of the Third International Workshop on Web Services and Formal Methods*, September 2006.

[Bra00] S. A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy.* The MIT Press, 2000.

[BRC+07] Sonia Ben Mokhtar, Pierre-Guillaume Raverdy, Roberto Speicys Cardoso, Aitor Urbieta, and Nikolaos Georgantas. Discovering Social Services in Pervasive Environments with Privacy – Demonstration Paper. In *ACM/IFIP/USENIX 8th International Middleware Conference (Middleware 2007)*, November 2007.

[Bri67] R. C. Brictson. Some Thoughts on the Social Implications of Computers and Privacy. Technical Report SP-2953, System Development Corp., 1967.

[Bri96] David Brin. The Transparent Society. *Wired Magazine*, 4(12), 1996. Available at `www.wired.com/wired/archive/4.12/fftransparent_pr.html`.

[Bri99] David Brin. *The Transparent Society: Will Technology Force us to Choose Between Privacy and Freedom*. Basic Books, 1999.

[BRUC08] Sonia Ben Mokhtar, Pierre-Guillaume Raverdy, Aitor Urbieta, and Roberto Speicys Cardoso. Interoperable Semantic & Syntactic Service Matching for Ambient Computing Environments. In *Proceedings of the 1st International Workshop on Ad-hoc Ambient Computing*, September 2008.

[BS03] A. R. Beresford and F. Stajano. Location Privacy in Pervasive Computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.

[C+08] Mauro Caporuscio et al. *Deliverable D3.3 - Middleware: Assessment and Revision*. IST PLASTIC Project, March 2008.

[Cam08] Kim Cameron. Are Countrywide's Systems Designed Around Need to Know?, October 2008. Available at `www.identityblog.com/?p=1013`.

[Cap04] Licia Capra. Engineering Human Trust in Mobile System Collaborations. In *Proceedings of the 12th International Symposium on the Foundations of Software Engineering*, November 2004.

[Cat00] Jason Catlett. Open Letter to P3P Developers & Replies. In *CFP '00: Proceedings of the tenth conference on Computers, Freedom and Privacy*, pages 157–164. ACM Press, 2000.

[CBM+02] Licia Capra, Gordon S. Blair, Cecilia Mascolo, Wolfgang Emmerich, and Paul Grace. Exploiting Reflection in Mobile Computing Middleware. *SIGMOBILE Mobile Comput. Commun. Rev.*, 6(4):34–44, 2002.

[CBUI07] Roberto Speicys Cardoso, Sonia Ben-Mokhtar, Aitor Urbieta, and Valérie Issarny. EVEY: Enhancing Privacy of Service Discovery in Pervasive Computing. In *ACM/IFIP/USENIX 8th International Middleware Conference (Middleware 2007)*, November 2007.

[CC08] Roberto Speicys Cardoso and Mauro Caporuscio. Exploring Multi-path Communication in Hybrid Wireless Ad Hoc Networks. In *1st International Workshop on Ad-hoc Ambient Computing*, September 2008.

[CCB00] J. L. Crowley, J. Coutaz, and F. Bérard. Perceptual User Interfaces: Things that See. *Communications of the ACM*, 43(3):54–64, 2000.

[CCMN05] G. Chafle, S. Chandra, V. Mann, and M. G. Nanda. Orchestrating Composite Web Services under Data Flow Constraints. In *ICWS '05: Proceedings of the IEEE International Conference on Web Services*, 2005.

[CDK+02] Francisco Curbera, Matthew Duftler, Rania Khalaf, William Nagy, Nirmal Mukhi, and Sanjiva Weerawarana. Unraveling the Web Services Web: An Introduction to SOAP, WSDL, and UDDI. *IEEE Internet Computing*, 6(2), March 2002.

[CF96] C. Carlsson and R. Fullér. Adaptive Fuzzy Cognitive Maps for Hyperknowledge Representation in Strategy Formation Process. In *Proceedings of International Panel Conference on Soft and Intelligent Computing*, 1996.

[CF03] I. Constantinescu and B. Faltings. Efficient Matchmaking and Directory Services. In *WI '03: Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence*, 2003.

[CFH06] Barbara Carminati, Elena Ferrari, and Patrick C. K. Hung. Security Conscious Web Service Composition. In *ICWS '06: Proceedings of the IEEE International Conference on Web Services*, 2006.

[CG98] J. Cohen and A. S. Gutterman. *Trade Secrets Protection and Exploitation.* BNA Books, 1998.

[CGKS95] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private Information Retrieval. In *FOCS'95: Proceedings of the 36th Annual IEEE Conference on Foundations of Computer Science*. IEEE Computer Society, 1995.

[CH91] D. Chaum and E. Van Heyst. Group Signatures. In *Eurocrypt '91*, pages 257–265, 1991.

[Cha81] D. L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[Cha88] D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[CHL+04] E. S. Chung, J. I. Hong, J. Lin, M. K. Prabaker, J. A. Landay, and A. L. Liu. Development and Evaluation of Emerging Design Patterns for Ubiquitous Computing. In *DIS '04:Proceedings of the 5th conference on Designing interactive systems: processes, practices, methods, and techniques*, pages 233–242, 2004.

[CI07] R. S. Cardoso and V. Issarny. Architecting Pervasive Computing Systems for Privacy: A Survey. In *Working IEEE/IFIP Conference on Software Architecture (WICSA '07)*, 2007.

[CIC08] Damien Charlet, Valérie Issarny, and Rafik Chibout. Energy-efficient Middleware-layer Multi-radio Networking: An Assessment in the Area of Service Discovery. *Computer Networks*, 52(1), 2008.

[Cla99] Roger Clarke. Internet Privacy Concerns Confirm the Case for Intervention. *Communications of the ACM*, 42(2):60–67, 1999.

[CLM02] Lorrie Cranor, Marc Langheinrich, and Massimo Marchiori. A P3P Preference Exchange Language 1.0 (APPEL1.0). W3C Working Draft, April 2002. Available at `www.w3.org/TR/P3P-preferences`.

[Coc07] Martin Cochran. Notes on the Wang et al. 2^{63} SHA-1 Differential Path. Cryptology ePrint Archive, Report 2007/474, 2007. `eprint.iacr.org`.

[Coh92] J. L. Cohen. Redescribing Privacy: Identity, Difference, and the Abortion Controversy. *Columbia Journal of Gender and Law*, 43(3), 1992.

[Coh03a] Bram Cohen. Incentives Build Robustness in BitTorrent. In *First Workshop on Economics of Peer-to-Peer Systems*, pages 116–121, 2003.

[Coh03b] Julie E. Cohen. DRM and Privacy. *Communications of the ACM*, 46(4):46–49, 2003.

[Con05] PRIME Consortium. Privacy and Identity Management for Europe - PRIME White Paper V1.0, 2005. Available at `www.prime-project.eu.org/public/press_room/whitepaper/PRIME-Whitepaper-V1.pdf`.

[Con06] Consumers Union of the United States Inc. Consumer Report Investigates: Your Privacy for Sale. *Consumer Reports*, 71(10), October 2006.

[CPA+08] Ashish Choudhary, Arpita Patra, B. V. Ashwinkumar, K. Srinathan, and C. Pandu Rangan. *Information Theoretic Security*, chapter Perfectly Reliable and Secure Communication Tolerating Static and Mobile Mixed Adversary. Springer Berlin/Heidelberg, 2008.

[Cra02] Lorrie Faith Cranor. *Web Privacy with P3P*. O'Reilly Media, Inc, 1st edition, September 2002.

[Csi97] László Csirmaz. The Size of a Share Must Be Large. *Journal of Cryptology*, 10(4), 1997.

[CT99] J. P. Carvalho and J. A. B. Tomé. Rule Based Fuzzy Cognitive Maps: Fuzzy Causal Relations. In *CIMCA'99: The 1999 International Conference on Computational Intelligence for Modelling, Control and Automation*, 1999.

[CT02] J. P. Carvalho and J. A. B. Tomé. Issues on the Stability of Fuzzy Cognitive Maps and Rule-based Fuzzy Cognitive Maps. In *Proceedings of the 2002 Annual Meeting of the North American Fuzzy Information Processing Society*, 2002.

[CZH+99] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz. An Architecture for a Secure Service Discovery Service. In *MobiCom '99: 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1999.

[DDM03] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *SP '03: Proceedings of the 2003 IEEE Symposium on Security and Privacy*, May 2003.

[DDWY93] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly Secure Message Transmission. *Journal of the ACM*, 40(1), 1993.

[Den80] Dorothy E. Denning. Secure Statistical Databases with Random Sample Queries. *ACM Transactions on Database Systems*, 5(3):291–315, 1980.

[DH76] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[DK93] J. A. Dickerson and B. Kosko. Virtual Worlds as Fuzzy Cognitive Maps. In *Proceedings of the IEEE Virtual Reality Annual International Symposium*, 1993.

[DK05] M. Duckham and L. Kulik. A Formal Model of Obfuscation and Negotiation for Location Privacy. In *Proceedings of PERVASIVE 2005*, pages 152–170, 2005.

[DMS04] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *Proceedings of the 13th USENIX*, August 2004.

[Dru69] Peter F. Drucker. *The Age of Discontinuity: Guidelines to Our Changing Society*. HarperCollins Publishers, 1969.

[DSNW+99] Y. Desmedt, R. Safavi-Naini, H. Wang, C. Charnes, and J. Pieprzyk. Broadcast Anti-Jamming Systems. In *ICON '99: Proceedings of the 7th IEEE International Conference on Networks*. IEEE Computer Society, 1999.

[DWB06] Yvo Desmedt, Yongge Wang, and Mike Burmester. *Critical Information Infrastructures Security*, chapter Revisiting Colored Networks and Privacy Preserving Censorship. Springer Berlin/Heidelberg, 2006.

[EJ01] D. Eastlake, 3rd and P. Jones. RFC3174: US Secure Hash Algorithm 1 (SHA1), September 2001. Available at `www.ietf.org/rfc/rfc3174.txt`.

[Emm00a] Wolfgang Emmerich. *Engineering Distributed Objects*. Wiley, 1st edition, June 2000.

[Emm00b] Wolfgang Emmerich. Software Engineering and Middleware: a Roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, 2000.

[EP06] Electronic Privacy Information Center and Privacy International, editors. *Privacy and Human Rights 2006 - An International Survey of Privacy Laws and Developments*. Electronic Privacy Information Center, 2006.

[FL07] J. Farrell and H. Lausen. Semantic Annotations for WSDL and XML Schema. W3C Working Draft, 2007.

[Fla89] D. H. Flaherty. *Protecting Privacy in Surveillance Societies*. University of North Carolina Press, 1989.

[FM02] M. J. Freedman and R. Morris. Tarzan: a peer-to-peer anonymizing network layer. In *CCS '02: Proceedings of the 9th ACM conference on Computer and communications security*, pages 193–206, November 2002.

[Fon08] John Fontana. Microsoft Says Credentica Acquisition will Help Users Protect Privacy. *Network World*, July 2008. Available at `www.networkworld.com/news/2008/030708-microsoft-credentica.html`.

[FPC03] R. Falcone, G. Pezzulo, and C. Castelfranchi. A Fuzzy Approach to a Belief-Based Trust Computation. In *Trust, Reputation, and Security: Theories and Practice*, 2003.

[Fri84] Charles Fried. *Philosophical Dimensions of Privacy*, chapter Privacy (a Moral Analysis). Cambridge University Press, 1984.

[FW00] Matthew Franklin and Rebecca N. Wright. Secure Communication in Minimal Connectivity Models. *Journal of Cryptology*, 13(1), September 2000.

[FZ94] George H. Forman and John Zahorjan. The Challenges of Mobile Computing. *Computer*, 27(4), 1994.

[Gar00] Simson Garfinkel. *Database Nation: the Death of Privacy in the 21st Century*. O'Reilly & Associates, Inc., 2000.

[Gas04] William Gasarch. A Survey on Private Information Retrieval. *Bulletin of the EATCS*, 82:72–107, 2004.

[GDS01] N. Golmie, R. E. Van Dyck, and A. Soltanian. Interference of Bluetooth and IEEE 802.11: Simulation Modeling and Performance Evaluation. In *MSWIM '01: Proceedings of the 4th ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems*, 2001.

[GG03] M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *ACM MobiSys2003*, pages 31–42, 2003.

[GIKM98] Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting Data Privacy in Private Information Retrieval Schemes. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 151–160, 1998.

[Gol06] P. Golle. Revisiting the Uniqueness of Simple Demographics in the U.S. Population. In *WPES '06: Proceedings of the 5th ACM Workshop on Privacy in Electronic Society*, 2006.

[Gou89] Gouvernement de la France. Déclaration des Droits de l'Homme et du Citoyen, 1789. Available at `www.conseil-constitutionnel.fr/textes/d1789.htm`.

[Gov88] Government of the U.S.A. The Constitution of the United States, 1788. Available at `www.archives.gov/exhibits/charters/constitution.html`.

[Gri04] Robert Grimm. One.world: Experiences with a Pervasive Computing Architecture. *IEEE Pervasive Computing*, 3(3), 2004.

[GRPS03] Sharad Goel, Mark Robson, Milo Polte, and Emin Gun Sirer. Herbivore: A Scalable and Efficient Protocol for Anonymous Communication. Technical Report 2003-1890, Cornell University, February 2003.

[GRS99] David Goldschlag, Michael Reed, and Paul Syverson. Onion Routing for Anonymous and Private Internet Connections. *Communications of the ACM*, 42(2), February 1999.

[GT96] Ceki Gulcu and Gene Tsudik. Mixing Email with Babel. In *SNDSS '96: Proceedings of the 1996 Symposium on Network and Distributed System Security*, 1996.

[GWB97] I. Goldberg, D. Wagner, and E. Brewer. Privacy-Enhancing Technologies for the Internet. In *COMPCON '97: Proceedings of the 42nd IEEE International Computer Conference*, 1997.

[Hag92] M. Hagiwara. Extended Fuzzy Cognitive Maps. In *IEEE International Conference on Fuzzy Systems*, 1992.

[Hay05] Richard Haynes. *Media Rights And Intellectual Property*. Edinburgh University Press, 2005.

[HB03] Rachid Hamadi and Boualem Benatallah. A Petri Net-based Model for Web Service Composition. In *ADC '03: Proceedings of the 14th Australasian database conference*, 2003.

[HL04] J. I. Hong and J. A. Landay. An Architecture for Privacy-Sensitive Ubiquitous Computing. In *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 177–189, 2004.

[HM97] Martin Hirt and Ueli Maurer. Complete Characterization of Adversaries Tolerable in Secure Multi-party Computation. In *PODC '97: Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing*, 1997.

[HNLL04] Jason I. Hong, Jennifer D. Ng, Scott Lederer, and James A. Landay. Privacy Risk Models for Designing Privacy-Sensitive Ubiquitous Computing Systems. In *DIS '04: Proceedings of the 2004 conference on Designing interactive systems*, pages 91–100. ACM Press, August 2004.

[HS96] S. E. Hudson and I. Smith. Techniques for Addressing Fundamental Privacy and Disruption Tradeoffs in Awareness Support Systems. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work*, pages 248–257, 1996.

[IP99] Lucas D. Introna and Athanasia Pouloudi. Privacy in the Information Age: Stakeholders, Interests and Values. *Journal of Business Ethics*, 22:27–38, 1999.

[IST+05] Valérie Issarny, Daniele Sacchetti, Ferda Tartanoglu, Françoise Sailhan, Rafik Chibout, Nicole Levy, and Angel Talamona. Developing Ambient Intelligence Systems: A Solution based on Web Services. *Automated Software Engineering*, 12(1), January 2005.

[IY04] K. Irwin and T. Yu. An Identifiability-Based Access Control Model for Privacy Protection in Open Systems. In *WPES '04: Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society*, 2004.

[JMB01] David B. Johnson, David A. Maltz, and Josh Broch. *Ad hoc Networking*, chapter DSR: the Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks, pages 139–172. Addison-Wesley Longman Publishing Co., Inc., 2001.

[JMC+01] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized Link State Routing Protocol for Ad Hoc Networks. In *INMIC 2001: Proceedings of the IEEE International Multi Topic Conference*, 2001.

[JW03] M. Jeronimo and J. Weast. *UPnP Design by Example*. Intel Press, May 2003.

[KCBC02] Fabio Kon, Fabio Costa, Gordon Blair, and Roy H. Campbell. The case for reflective middleware. *Communications of the ACM*, 45(6):33–38, 2002.

[KCRI09] Natallia Kokash, Roberto Speicys Cardoso, Pierre-Guillaume Raverdy, and Valérie Issarny. A flexible qos-aware routing protocol for infrastructureless b3g networks. In *Proceedings of the 24th Annual ACM Symposium on Applied Computing - Mobile Computing and Applications Track*, March 2009.

[KGSR02] M V N Ashwin Kumar, Pranava R. Goundan, K Srinathan, and C. Pandu Rangan. On Perfectly Secure Communication over Arbitrary Networks. In *PODC '02: Proceedings of the twenty-first annual symposium on Principles of distributed computing*, 2002.

[KK03] L. Korba and S. Kenny. Towards Meeting the Privacy Challenge: Adapting DRM. In *ACM CCS-9 Workshop*, pages 118–136, 2003.

[KLC04] I. Kang, S. Lee, and J. Choi. Using Fuzzy Cognitive Maps for the Relationship Management in Airline Service. *Expert Systems with Applications*, 26(4), 2004.

[Koh78] Loren M. Kohnfelder. *Towards a Practical Public-key Cryptosystem.* PhD thesis, Massachusetts Institute of Technology, May 1978.

[Kos86] B. Kosko. Fuzzy Cognitive Maps. *International Journal of Man-Machine Studies*, 24(1), 1986.

[Kos91] B. Kosko. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intelligence.* Prentice-Hall International Editions, 1991.

[Kra94] Hugo Krawczyk. Secret Sharing Made Short. In *CRYPTO '93: Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology*, 1994.

[KS01] K. Knorr and H. Stormer. Modeling and Analyzing Separation of Duties in Workflow Environments. In *Sec '01: Proceedings of the 16th International Conference on Information Security: Trusted Information*, 2001.

[KVH97] A. Krall, J. Vitek, and N. Horspool. Near Optimal Hierarchical Encoding of Types. In *ECOOP'97: 11th European Conference on Object Oriented Programming*, 1997.

[KYS05] H. Kido, Y. Yanagisawa, and T. Satoh. An Anonymous Communication Technique using Dummies for Location-based Services. In *ICPS'05: Proceedings of the IEEE International Conference on Pervasive Services*, pages 88–97, 2005.

[Lam73] B. W. Lampson. A Note on the Confinement Problem. *Communications of the ACM*, 16(10):613–615, 1973.

[Lan01] M. Langheinrich. Privacy by Design - Principles of Privacy-Aware Ubiquitous Systems. In *UbiComp 2001: Proceedings of the 3rd international conference on Ubiquitous Computing*, pages 273–291, 2001.

[Lan02] M. Langheinrich. A Privacy Awareness System for Ubiquitous Computing Environments. In *UbiComp 2002: Proceedings of the 4th international conference on Ubiquitous Computing*, pages 237–245, 2002.

[Les04] Lawrence Lessig. *Free Culture: How Big Media Uses Technology and the Law to Lock Down Culture and Control Creativity.* The Penguin Press, 2004.

[Les06] L. Lessig. *Code - version 2.0.* Basic Books, 2006. Available at `codev2.cc`.

[LG00] Sung-Ju Lee and Mario Gerla. AODV-BR: Backup Routing in Ad Hoc Networks. In *WCNC'00: Wireless Communications and Networking Conference*, 2000.

[LG01] Sung-Ju Lee and Mario Gerla. Split Multipath Routing with Maximally Disjoint Paths in Ad hoc Networks. In *ICC 2001: IEEE International Conference on Communications*, 2001.

[LH04] Lei Li and Ian Horrocks. A Software Framework for Matchmaking Based on Semantic Web Technology. *International Journal of Electronic Commerce*, 8(4), 2004.

[LI04a] J. Liu and V. Issarny. Service Allocation in Selfish Mobile Ad hoc Networks Using Vickrey Auction. In *Current Trends in Database Technology - EDBT 2004 Workshops*, pages 385–394, 2004.

[LI04b] Jinshan Liu and Valérie Issarny. Enhanced Reputation Mechanism for Mobile Ad Hoc Networks. In *Proceedings of the Second iTrust International Conference*, March 2004.

[LJ04] S. Lahlou and F. Jegou. European Disappearing Computer Privacy Design Guidelines V1.1. Technical Report EDC-PG 2003, Ambient Agoras, 2004.

[LLF04] Wenjing Lou, Wei Liu, and Yuguang Fang. SPREAD: Enhancing Data Confidentiality in Mobile Ad Hoc Networks. In *INFOCOM 2004: Proceedings of the Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, 2004.

[LLR05] Saadi Lahlou, Marc Langheinrich, and Carsten Röcker. Privacy and Trust Issues with Invisible Computers. *Communications of the ACM*, 48(3), 2005.

[LM07] Roberto Lucchi and Manuel Mazzara. A pi-calculus Based Semantics for WS-BPEL. *Journal of Logic and Algebraic Programming*, 70(1), 2007.

[LSE03] D. Davide Lamanna, James Skene, and Wolfgang Emmerich. SLAng: A Language for Defining Service Level Agreements. In *FTDCS '03: Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems*, 2003.

[LU02] Joe Loughry and David A. Umphress. Information Leakage from Optical Emanations. *ACM Transactions on Information and System Security*, 5(3), 2002.

[Man01] Steve Mann. Guest Editor's Introduction: Wearable Computing-Toward Humanistic Intelligence. *IEEE Intelligent Systems*, 16(3), 2001.

[Mar01] Gary T. Marx. Murky Conceptual Waters: The Public and the Private. *Ethics and Information Technology*, 3(3):157–169, 2001.

[McH95] John McHugh. *Handbook for the Computer Security Certification of Trusted Systems*, chapter Covert Channel Analysis. Naval Research Laboratory, 1995.

[MD01] Mahesh K. Marina and Samir R. Das. On-demand Multipath Distance Vector Routing in Ad Hoc Networks. In *Ninth International Conference on Network Protocols*, 2001.

[MD05] Carlos Aguilar Melchor and Yves Deswarte. pMIX: Untraceability for Small Hiding Groups. In *NCA '05: Proceedings of the Fourth IEEE International Symposium on Network Computing and Applications*, pages 29–40, July 2005.

[MD06] Carlos Aguilar Melchor and Yves Deswarte. From DC-Nets to pMIXes: Multiple Variants for Anonymous Communications. In *NCA '06: Proceedings of the Fifth IEEE International Symposium on Network Computing and Applications*, pages 163–172, 2006.

[Mil93] R. Milner. The Polyadic π-Calculus: A Tutorial. In Friedrich Ludwig Bauer, W. Brauer, and H. Schwichtenberg, editors, *Logic and Algebra of Specification*, 1993.

[Mit95] William J. Mitchell. *City of Bits: Space, Place, and the Infobahn.* MIT Press, 1995.

[MM04] Nikola Milanovic and Miroslaw Malek. Current Solutions for Web Service Composition. *IEEE Internet Computing*, 8(6), 2004.

[MN03] M. McCahill and C. Norris. CCTV Systems in London - Their Structure and Practices. Technical Report 10, Urbaneye Project, 2003.

[MPG+08] Sonia Ben Mokhtar, Davy Preuveneers, Nikolaos Georgantas, Valérie Issarny, and Yolande Berbers. EASY: Efficient semAntic Service discoverY in pervasive computing environments with QoS and context support. *Journal of Systems and Software*, 81(5), 2008.

[MSLL99] Yuan Miao, Chee Kheong Siew, Zhi-Qiang Liu, and Shi Li. Dynamical Cognitive Network An Extension of Fuzzy Cognitive Map. In *ICTAI '99: Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence*, 1999.

[Mur07] Steven J. Murdoch. *Covert Channel Vulnerabilities in Anonymity Systems.* PhD thesis, University of Cambridge, December 2007.

[Neb05] Pierre Nebel. Vous Avez la Carte Mouchard? *L'Hebdo*, July 2005. Available at `www.hebdo.ch/index.cfm?id=1841`.

[NG03] C. Neustaedter and S. Greenberg. The Design of a Context-Aware Home Media Space for Balancing Privacy and Awareness. In *UbiComp 2003*, pages 297–314, 2003.

[NM06] D.-Q. Nguyen and P. Minet. QoS Support and OLSR Routing in a Mobile ad hoc Network. In *ICNICONSMCL '06: Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies*, 2006.

[OAS05a] OASIS. *Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005. Available at `docs.oasis-open.org/security/saml/v2.0`.

[OAS05b] OASIS. *Profiles for the OASIS Security Assertion Markup Language (SAML) V2.0*, March 2005. Available at `docs.oasis-open.org/security/saml/v2.0/`.

[OAS05c] OASIS. *Security Assertion Markup Language (SAML) V2.0*, March 2005. Available at `saml.xml.org/saml-specifications`.

[OAS05d] OASIS. *UDDI Version 3.0*, February 2005. Available at `www.oasis-open.org/committees/uddi-spec/doc/tcspecs.htm#uddiv3`.

[OAS07] OASIS. *Web Services Business Process Execution Language Version 2.0*, April 2007. Available at `docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html`.

[O'H05] R. O'Harrow, Jr. *No Place to Hide*. Free Press, 2005.

[Org80] Organisation for Economic Co-operation and Development (OECD). Recommendation of the Council Concerning Guidelines Governing the Protection of Privacy and Transborder Flows of Personal Data, September 1980.

[Orw50] George Orwell. *1984*. Signet Classics, 1950.

[PB06] D. Preuveneers and Y. Berbers. Prime Numbers Considered Useful: Ontology Encoding for Efficient Subsumption Testing. Technical Report CW464, Department of Computer Science, Katholieke Universiteit Leuven, Belgium, 2006.

[PC97] Vincent D. Park and M. Scott Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *INFOCOM '97: Proceedings of the Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution*. IEEE Computer Society, 1997.

[Per96] K. Perusich. Fuzzy Cognitive Maps for Policy Analysis. In *Proceedings of the International Symposium on Technology and Society Technical Expertise and Public Decisions*, 1996.

[PG03] M. P. Papazoglou and D. Georgakopoulos. Service-Oriented Computing. *Communications of the ACM*, 46(10), October 2003.

[PHB06] Alexander Pretschner, Manuel Hilty, and David Basin. Distributed Usage Control. *Communications of the ACM*, 49(9), 2006.

[PKPS02] M. Paolucci, T. Kawamura, T. R. Payne, and K. P. Sycara. Semantic Matching of Web Services Capabilities. In *ISWC '02: First International Semantic Web Conference*, 2002.

[PR99] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc On-Demand Distance Vector Routing. In *WMCSA '99: Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications*, 1999.

[PSC+07] Arpita Patra, Bhavani Shankar, Ashish Choudhary, K. Srinathan, and C. Pandu Rangan. Perfectly Secure Message Transmission in Directed Networks Tolerating Threshold and Non Threshold Adversary. In *CANS 2007: Proceedings of the 6th International Conference on Cryptology and Network Security*, 2007.

[PSG06] Elpiniki I. Papageorgiou, Chrysostomos Stylios, and Peter P. Groumpos. Unsupervised Learning Techniques for Fine-tuning Fuzzy Cognitive Map Causal Links. *International Journal of Human-Computer Studies*, 64(8):727–743, 2006.

[PTDL07] Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-Oriented Computing: State of the Art and Research Challenges. *Computer*, 40(11), 2007.

[Rab89] Michael O. Rabin. Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance. *Journal of the ACM*, 36(2), 1989.

[RAI06] P.-G. Raverdy, S. Armand, and V. Issarny. Scalability Study of the MUSDAC Platform for Service Discovery in B3G Networks. In *WWRF-17: Proceedings of Wireless World Research Forum Meeting*, November 2006.

[Ran04] K. Ranganathan. Trustworthy Pervasive Computing: the Hard Security Problems. In *Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops*, March 2004.

[RC99] J. Reagle and L. F. Cranor. The Platform for Privacy Preferences. *Commununications of the ACM*, 42(2):48–55, 1999.

[RFIG07] L. Rong, M. Fredj, V. Issarny, and N. Georgantas. Mobility Management in B3G Networks: a Middleware-based Approach. In *ESSPE '07: International Workshop on Engineering of Software Services for Pervasive Environments*, 2007.

[RHC+02] Manuel Román, Christopher Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, and Klara Nahrstedt. A Middleware Infrastructure for Active Spaces. *IEEE Pervasive Computing*, 1(4), 2002.

[RKC01] Manuel Roman, Fabio Kon, and Roy H. Campbell. Reflective Middleware: From Your Desk to Your Hand. *IEEE Distributed Systems Online*, 2(5), 2001.

[RP02] Marc Rennhard and Bernhard Plattner. Introducing MorphMix: Peer-to-peer Based Anonymous Internet Usage with Collusion Detection. In *WPES '02: Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society*, November 2002.

[RPM00] Gruia-Catalin Roman, Gian Pietro Picco, and Amy L. Murphy. Software Engineering for Mobility: a Roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, 2000.

[RR98] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[RRdLC+06] P.-G. Raverdy, O. Riva, A. de La Chapelle, R. Chibout, and V. Issarny. Efficient Context-aware Service Discovery in Multi-Protocol Pervasive Environments. In *MDM'06: Proceedings of the 7th International Conference on Mobile Data Management*, May 2006.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Communications of the ACM*, 21(2), February 1978.

[RSG98] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous Connections and Onion Routing. *IEEE Journal on Selected Areas in Communications*, 16(4), May 1998.

[RST01] R. L. Rivest, A. S., and Y. Tauman. How to Leak a Secret. In *Proceedings of ASIACRYPT '01*, pages 552–565, 2001.

[Sat01] M. Satyanarayanan. Pervasive Computing: Vision and Challenges. *IEEE Personal Communications*, 8(4), August 2001.

[Sav05] Richard Savill. Magistrate Fined for Keeping Lost Rolex. *Telegraph*, January 2005. Available at `www.telegraph.co.uk/news/main.jhtml?xml=/news/2005/01/28/nrolex28.xml&sSheet=/news/2005/01/28/ixhome.html`.

[SBS02] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P5: A Protocol for Scalable Anonymous Communication. In *SP '02: Proceedings of IEEE Security and Privacy Symposium*, 2002.

[SC03] Frank Stajano and Jon Crowcroft. *Ambient Intelligence: Impact on Embedded System Design*, chapter The Butt of the Iceberg: Hidden Security Problems of Ubiquitous Systems. Springer US, 2003.

[SC07] Radu Sion and Bogdan Carbunar. On the Practicality of Private Information Retrieval. In *NDSS'07: Proceedings of the 14th Annual Network & Distributed System Security Symposium*, 2007.

[Sch77] Carl D. Schneider. *Shame, Exposure, and Privacy.* W W Norton & Co, 1977.

[Sch96] Bruce Schneier. *Applied Cryptography.* John Wiley & Sons, 1996.

[Sch06] B. Schneier. The Eternal Value of Privacy. Wired News, May 2006. Available at `www.schneier.com/essay-114.html`.

[Sch08] Bruce Schneier. The Myth of the 'Transparent Society'. Wired Commentary - Security Matters, 2008. Available at `www.wired.com/politics/security/commentary/securitymatters/2008/03/securitymatters_0306`.

[SG99] Chrysostomos D. Stylios and Peter P. Groumpos. Fuzzy Cognitive Maps: A Model for Intelligent Supervisory Control Systems. *Computers in Industry*, 39(3):229–238, 1999.

[SG00] C. D. Stylios and P. P. Groumpos. Fuzzy Cognitive Maps in Modeling Supervisory Control Systems. *Journal of Intelligent and Fuzzy Systems*, 8(2), 2000.

[SGG97] Chrysostomos D. Stylios, Voula C. Georgopoulos, and Peter P. Groumpos. The Use of Fuzzy Cognitive Maps in Modeling Systems. In *Proceedings of the 5th IEEE Mediterranean Conference on Control and Systems*, July 1997.

[SH05] M. P. Singh and M. N. Huhns. *Service-Oriented Computing - Semantics, Processes, Agents.* John Wiley and Sons, September 2005.

[Sha48] C. E. Shannon. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27, 1948.

[Sha79] Adi Shamir. How to Share a Secret. *Communications of the ACM*, 22(11), 1979.

[SHP03] E. Sirin, J. Hendler, and B. Parsia. Semi-automatic Composition of Web Services Using Semantic Descriptions. In *Web Services: Modeling, Architecture and Infrastructure Workshop in conjunction with ICEIS 2003*, 2003.

[SI05] F. Sailhan and V. Issarny. Scalable Service Discovery for MANET. In *PERCOM '05: 3rd IEEE International Conference on Pervasive Computing and Communications*, 2005.

[SKR02] K. Srinathan, M. V. N. Ashwin Kumar, and C. Pandu Rangan. Asynchronous Secure Communication Tolerating Mixed Adversaries. In *ASIACRYPT '02: Proceedings of the 8th International Conference on the Theory and Application of Cryptology and Information Security*. Springer-Verlag, 2002.

[Spr04] Brandon Sprague. Fireman Attempted to Set Fire to House, Charges Say. *The Seattle Times*, October 2004. Available at `seattletimes.nwsource.com/html/localnews/2002055245_arson06m.html`.

[SS73] Jerome H. Saltzer and Michael D. Schroeder. The Protection of Information in Computing Systems. In *SOSP '73: Proceedings of the Fourth ACM Symposium on Operating Systems Principles*, pages 1278– 1308. ACM Press, 1973.

[SS05] Andrei Serjantov and Peter Sewell. Passive-Attack Analysis for Connection-based Anonymity Systems. *International Journal of Information Security*, 4(3), June 2005.

[Str04] Lior Jacob Strahilevitz. A Social Networks Theory of Privacy. Technical report, The Law School - The University of Chicago, 2004.

[Swe00] L. Sweeney. Uniqueness of Simple Demographics in the U.S. Population. Technical Report LIDAP- WP4, Carnegie Mellon University, Laboratory for International Data Privacy, Pittsburgh, PA, 2000.

[SYKW99] Katia Sycara, Jianguo Lu Y, Matthias Klusch, and Seth Wido. Matchmaking among Heterogeneous Agents on the Internet. In *AAAI Spring Symposium on Intelligent Agents in Cyberspace*, 1999.

[TBG01] David Trastour, Claudio Bartolini, and Javier Gonzalez-Castillo. A Semantic Web Approach to Service Description for Matchmaking of Services. In *SWWS '01: Proceedings of the International Semantic Web Working Symposium*, 2001.

[The07] The Associated Press. TJX Says Theft of Credit Data Involved 45.7 Million Cards, March 2007. Available at `www.nytimes.com/2007/03/30/business/30data.html`.

[TYW84] J. F. Traub, Y. Yemini, and H. Woźniakowski. The Statistical Security of a Statistical Database. *ACM Transactions on Database Systems*, 9(4):672–679, 1984.

[Uni48] United Nations. Universal Declaration of Human Rights, 1948. Available at `www.un.org/Overview/rights.html`.

[Vin02] Steve Vinoski. Where is Middleware? *IEEE Internet Computing*, 6(2), 2002.

[VT05] W. M. P. Van der Aalst and A. H. M. Ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4), 2005.

[W3C01] W3C. *Web Services Description Language (WSDL) 1.1*, March 2001. Available at `www.w3.org/TR/wsdl`.

[W3C07] W3C. *Simple Object Access Protocol (SOAP) Version 1.2*, April 2007. Available at `www.w3.org/TR/soap`.

[Wal99] Jim Waldo. The Jini Architecture for Network-centric Computing. *Communications of the ACM*, 42(7):76–82, 1999.

[Wat06] Paul Watson. U. S. Military Secrets for Sale at Afghan Bazaar. *Los Angeles Times*, April 2006. Available at `articles.latimes.com/2006/apr/10/world/fg-disks10`.

[WB90] S. D. Warren and L. D. Brandeis. The Right to Privacy. *Harvard Law Review*, IV(5), 1890.

[WD02] Yongge Wang and Yvo Desmedt. Perfectly Secure Message Transmission Revisited. In *EUROCRYPT '02: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, 2002.

[Wei91] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265(3), September 1991.

[Wei93] Mark Weiser. Some Computer Science Issues in Ubiquitous Computing. *Communications of the ACM*, 36(7), July 1993.

[Wes67] Alan F. Westin. *Privacy and Freedom.* Atheneum Books, 1967.

[Wes72] Alan F. Westin. *Databanks in a Free Society: Computers, Record Keeping and Privacy.* Times Books, 1972.

[WGB99] M. Weiser, R. Gold, and J. S. Brown. The Origins of Ubiquitous Computing Research at PARC in the Late 1980s. *IBM Systems Journal*, 38(4), May 1999.

[Wie98] M. J. Wiener. Performance Comparison of Public-Key Cryptosystems. *RSA Laboratories' CryptoBytes*, 4(1), July 1998.

[WLLP01] Brett Warneke, Matt Last, Brian Liebowitz, and Kristofer S. J. Pister. Smart Dust: Communicating with a Cubic-Millimeter Computer. *Computer*, 34(1), 2001.

[WS08] Peter Williams and Radu Sion. Usable PIR. In *NDSS'08: Proceedings of the 15th Annual Network & Distributed System Security Symposium*, 2008.

[WSA+96] Roy Want, Bill N. Schilit, Norman I. Adams, Rich Gold, Karin Petersen, David Goldberg, John R. Ellis, and Mark Weiser. *Mobile Computing*, chapter The Parctab Ubiquitous Computing Experiment. The Springer International Series in Engineering and Computer Science. Springer US, 1996.

[WZSD00] Lei Wang, Lianfang Zhang, Yantai Shu, and Miao Dong. Multipath Source Routing in Wireless Ad hoc Networks. In *Proceedings of the Canadian Conference on Electrical and Computer Engineering*, 2000.

[YBCD08] Jin Yu, Boualem Benatallah, Fabio Casati, and Florian Daniel. Understanding Mashup Development. *IEEE Internet Computing*, 12(5), 2008.

[ZBN+04] Liangzhao Zeng, Boualem Benatallah, Anne H.H. Ngu, Marlon Dumas, Jayant Kalagnanam, and Henry Chang. QoS-Aware Middleware for Web Services Composition. *IEEE Transactions on Software Engineering*, 30(5), 2004.

[ZH99] Lidong Zhou and Zygmunt J. Haas. Securing Ad Hoc Networks. *IEEE Network*, 13(6), November 1999.

[Zim96] Philip Zimmermann. Testimony of Philip R. Zimmermann to the Subcommittee on Science, Technology, and Space of the US Senate Committee on Commerce, Science, and Transportation, June 1996.

[ZMN04] F. Zhu, M. Mutka, and L. Ni. PrudentExposure: A Private and User-centric Service Discovery Protocol. In *PERCOM '04: 2nd IEEE International Conference on Pervasive Computing and Communications*, 2004.

[ZMN05] Feng Zhu, Matt W. Mutka, and Lionel M. Ni. Service Discovery in Pervasive Computing Environments. *IEEE Pervasive Computing*, 4(4):81–90, 2005.

[ZS98] Q. A. Zhao and J. T. Stasko. Evaluating Image Filtering Based Techniques in Media Space Applications. In *Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 11–18, 1998.

[ZW97] Amy Moormann Zaremski and Jeannette M. Wing. Specification Matching of Software Components. *ACM Transactions on Software Engineering and Methodology*, 6(4), 1997.

[ZZMN05] F. Zhu, W. Zhu, M. W. Mutka, and L. Ni. Expose or Not? A Progressive Exposure Approach for Service Discovery in Pervasive Computing Environments. *PERCOM '05: 3rd IEEE International Conference on Pervasive Computing and Communications*, 2005.