TOWARD A *LOCALISATION OF TRUST* FRAMEWORK FOR PERVASIVE ENVIRONMENTS

Jun Li

School of Computer Science

Thesis submitted to
the University of Hertfordshire
in partial fulfilment of the requirements
of the degree of
Doctor of Philosophy
11 March, 2008.

Abstract

Pervasive computing envisions an environment in which we are surrounded by many embedded computer devices. The existence of those networked devices provides us with a mobile, spontaneous and dynamic way to access various resources provided by different (security policy) domains. In recent years, we have witnessed the evolutionary development of numerous *multiple domain* applications. One of the richest examples is pervasive environments. Typically, the conventional approach to secure access over multiple domains is to implement a unique trusted infrastructure, extending *local* identity or capability based security systems and combining them with *cross-domain* authentication mechanisms. However, this does not adequately meet the security requirements of communicating with unknown players in pervasive environments. Moreover, it is infeasible to define a global trust infrastructure and a set of assumptions that every player will *trust* in the multiple domain context. A powerful design technique to address those new security challenges posed by pervasive environments is to understand them from a *domain* perspective.

This thesis presents *Localisation of Trust* (LoT), an architectural framework designed to address the security need of how to talk to *correct* strangers in pervasive environments. Based on the *localising trust* security principle, LoT provides a generic platform for building access control over multiple domains from two ends: *authentication* and *authorisation*. Firstly, LoT proposes a two-channel authentication protocol to replace traditional (strong) identity-based authentication protocols by exploring desirable contextual information for different pervasive applications. Then, delegation and *localised* authentication are deployed to achieve authorisation in pervasive environments. The heart of this different semantic is to let the right domain get involved with its local players' interactions by helping them to convert a "token" to a usable

access capability, whilst keeping revocation in mind. This is done by introducing a domain-oriented Encryption-Based Access Control method, using ideas borrowed for Identity-based Encryption.

The second part of this thesis describes several specific mechanisms and protocols including a Dual Capabilities Model to achieve the required anti-properties for LoT. Although novel, they are intended primarily as an existence proof rather than being claimed to be ideal. Depending upon the precise application and context, other mechanisms may be better. Most importantly, the architecture-focused LoT provides such a flexibility by introducing multiple domains as a primary concern but leaving untouched the security protocols underlying each single domain and system implementation. Finally, a single domain scenario, guest access, is examined with the light of LoT. The purpose of doing so is to enhance the understanding of *domain* and other concepts described in LoT and demonstrate the effectiveness and efficiency of LoT for the scenarios chosen.

To my parents Si-De Huang and Xiao-Li Li

Acknowledgements

Foremost I would like to express my greatest appreciation to my supervisor, Prof. Bruce Christianson, for not only giving me this most wonderful opportunity to study for a Ph.D, but also for his continuous support, advice, guidance and tireless encouragement throughout the period of my research life. He has been exceptionally understanding, allowing me to challenge myself from different perspectives in addition to academic research. While completing my Ph.D, I finally understand what he told me in our first meeting, "Ph.D is a different animal". What I have learnt from my Ph.D research including commitment, dedication, self-motivation and the ability to keep intellectual curiosity are priceless.

I am also grateful to James Malcolm, who has provided me with constant critical discussion on this work regularly. Many thanks are also due to the fellow research students in the STRI to whom I have numerous intellectual, interesting and enjoyable discussions.

As always, I owe a special debt to my parents, Si-De Huang and Xiao-Li Li for not only allowing me to pursuit my dreams freely, but more importantly for their care, support and endless love.

Contents

Abstract **2**

Acknowledgements **5**

Contents **6**

List of Figures **10**

1 Introduction **11**

1.1 Single Domain vs. Multiple Domains 13

1.2 Definition 14

1.3 New Challenges 15

1.4 My Approach 18

1.5 Contribution 19

1.6 Dissertation Outline 21

2 Background **24**

2.1 Authentication 25

2.1.1 What Authentication Protocols Are 25

2.1.2 Authentication in Conventional Environments 27

2.1.3 Authentication On The Fly 29

2.1.4 Authentication 32

2.2 Authorisation 33

2.2.1 Access Control List 34

2.2.2 Role-based Access Control 34

2.2.3 Capability-based Access Control 35

2.3 Delegation 38

2.3.1 Certificate-based Delegation Schemes 39
2.3.2 Delegation of Responsibility 40
2.3.3 Trust and Policy Based Delegation Schemes 43
2.4 Revocation 45
2.4.1 The Principle of Trust 45
2.4.2 Various Revocation Schemes 46
2.5 A Short History of Public Key Cryptography 48
2.6 Conclusions 53

3 The Need for Localisation-of-Trust 54
3.1 Paradigm Shift in Security 54
3.2 Overview of the LoT Framework 57
3.3 The *Correct Strangers* Problem 61
3.4 Localising the Trust Security Principle 63
3.5 Security Countermeasures/Mechanisms 66
3.6 Conclusions 69

4 Spontaneous Authentication for Pervasive Environments 70
4.1 Basic Security Requirements in Pervasive Environments 71
4.1.1 Talking to Correct Strangers 71
4.1.2 Motivating Threat Model - Public Meeting 1 73
4.2 Significant Human Context in Pervasive Environments 75
4.2.1 Positive Human Context 75
4.2.2 Minimise The Reliance Upon Trustworthiness 77
4.3 Two Channel Authentication Protocols 78
4.3.1 Leave Strong Authentication Behind 78
4.3.2 Spontaneous Authentication with Two Channels Protocol 79
4.4 Example Protocols 82
4.4.1 Basic Approach 83
4.4.2 Generic Approach with two Channel Protocol 86

4.5 Conclusions 89

5 The Domain-Oriented Approach for Access Control Over Multiple Domains 90

5.1 Domain-Oriented Access Control Method 91

5.1.1 Access Control over Multiple Domains 91

5.1.2 Domain-Oriented Approach 93

5.2 Profile Certificates 96

5.3 Delegation and Access Control over Multiple Domains 100

5.3.1 Remote Authorisation - Delegation of Rights 101

5.3.2 Localised Authentication 102

5.4 Architecture for Encryption-based Access Control 105

5.5 Revocation 110

5.6 Conclusions 114

6 Semi-Trusted Agent Mediated Protocols for Access Control 115

6.1 The Overview of the Semi-Trusted Agent Mediated Protocol (STAMP) 116

6.2 Strangers Access from *Friendly* Domains 119

6.2.1 Motivating Threat Model - Public Meeting 2 119

6.2.2 Confining an Insider 120

6.3 Dual Capabilities Model (DuCaM) 121

6.4 Discrete-Logarithm based Scheme for Localised Authentication 124

6.5 The *Guest Access* Scenario 129

6.5.1 The Definition of *Guests* 130

6.5.2 Common Approaches for Guests Access 131

6.5.3 Motivating Threat Model - Guest Printing 132

6.5.4 Introducing Guests 133

6.6 Conclusions 135

7 Conclusions and Future Work **136**

7.1 Summary of Contributions 136

7.2 Directions for Future Work 137

7.2.1 Authentication and Expiration 138

7.2.2 Capability-based Access Control 139

7.2.3 Context-Awareness Applications in Pervasive Computing 141

7.2.4 Work that We Do Not Want To See 142

7.3 Conclusions 142

Bibliography **144**

List of Figures

2.1 Access Control List 34
2.2 Role-based Access Control 35
2.3 Capability-based Access Control 36
2.4 Public Key Distribution Problem 49
2.5 Public Key Infrastructure 50
2.6 Identity-based Cryptography 51

3.1 The LoT Framework Overview 59
3.2 Access For Local Players 61
3.3 Access For Remote Players 62
3.4 Triangle Trust Diagram 63
3.5 Trust over Multiple Domains 66

5.1 Trust Triangle 92
5.2 Access Across Domain Boundaries 94
5.3 Remote Authorisation - Different Delegation Semantics 103
5.4 Localised Authentication 104
5.5 The Overview of Encryption-based Access Control 107

6.1 STAMP Overview 117
6.2 Dual Capabilities 123
6.3 Localised Authentication in STAMP 126

Chapter 1

Introduction

As computing resources become increasingly pervasive, human users discover that they can access more and more resources spontaneously and dynamically. These resources in pervasive environments are usually provided by a variety of different suppliers, for instance, organisations, companies, universities, shops, even other human users, and so on. While much past research on security protocols for distributed systems has addressed the issues arising in conventional networking environments, today's spontaneous and highly decentralised pervasive environments applications have raised a number of new challenges in a considerably underexplored territory, *security over multiple domains*.

A number of security systems have been developed previously to apply traditional, well-studied schemes, including secret key exchange, public key certificates, access control lists and capabilities, etc., to satisfy the security needs of conventional environments (e.g. distributed systems). However, I am not aware of much work that attempts to clarify and focus on the security concern over multiple domains as the primary setting. In fact, most existing works target the security problems in a single domain first, then lift a security system (that is being successfully implemented in a single domain) to the multiple domain context. Usually, this is done by the implementation of a unique trusted infrastructure, extending (local) identity or capability based security

systems and combining them with cryptographic (cross-domain) authentication mechanisms. However, this approach will be cumbersome in pervasive environments due to the spontaneous and dynamic nature of pervasive interactions. More significantly, this is not what we really want. There is typically a lack of a global trust infrastructure which every player will trust. Moreover, it is infeasible to force each domain to implement the same (or even compatible) security infrastructure and mechanisms.

As a result, this dissertation presents my research on security over multiple domains, particularly the pervasive computing environment. I argue that security for the pervasive environment can only be partially solved, if security for multiple domains cannot be solved first. Based upon some current state-of-the-art of research works in the security filed, I am proposing a new security framework, called *LoT* (Localisation of Trust). *LoT* presents a conceptually *simple idea* [1] in spite of the considerably tricky protocol implementation. It is not a system implementation, as most underlying mechanisms adopted in LoT are not (purely) new. However, we intend to change the way of conventional thinking on security, particularly for pervasive environments. Consequently, in essence, LoT is an *architectural* security framework.

This chapter describes the motivation and outlines the contributions of this work. It begins with a brief discussion of *single* and *multiple domains*. This is followed by some definitions in section 1.2. Section 1.3 examines new challenges posed by multiple-domain oriented pervasive environments. Section 1.4 illustrates the approach that is adopted in this dissertation to guide the development of LoT. Section 1.5 outlines the contribution of this research. This chapter ends with the description of the structure of the rest of this dissertation.

[1] Although I have to admit that it took me three versions of draft to describe it *simply*.

1.1 Single Domain vs. Multiple Domains

The concept of *domain* is not entirely new in the security field, but may be used differently in different contexts. In conventional environments (e.g. distributed systems), *administrative domains* have been used to organise resources into *domains*. Often, the conventional administrative domains are based upon the geographic location (i.e. in the same machine, in the same building, attached to the same network) of resources for the purpose of easy administration and security.

Conversely, the concept of *domain* used in this dissertation is not based upon physical geography. Instead, it implicitly means a security (policy) domain which may span over several administrative domain boundaries (or geographic locations). As a consequence, if the same security policies apply to numerous (conventional) domains, those (conventional) domains are actually in a **single domain**. For example, two (administrative) domains (i.e. company A in London and company B in Cambridge) decide to implement the classic Kerberos [122] system and both reply upon one particular Kerberos server KS_e. From the security policy administration perspective, it is not surprising that both (administrative) domains are actually in the same (security) domain under domain server KS_e.

Thus, the term **multiple domains** will be used when resources are subject to different security policies, even though some of them may be residing in the same geographic location. The richest case of the multiple domain context is pervasive environments, when many players from different domains wish to interact with each other spontaneously and more openly. A concrete definition of the term *domain* will be given in the following section.

To access resources in a single domain, local players need to *contact* the domain servers, such as use of a pre-defined userID and password to log in, a shared key proof or a certificate. Those local players can be physically

located either within the geographical boundary of conventional domain, or outside the geographical boundary. Some attempts have been made to address the security issues in the later case recently, as witness mobile computing applications wishing to access resources over an unreliable link. From our viewpoint, this is not the big problem. Instead, the most central issue here is, *how to access resources in other domains*. This is much more difficult than to access resources in a user's own domain, because one player from a domain A has to struggle to understand another domain B's policy or its own players' roles. Hence, if an interaction involves a number of domains, it is hard to convince communicating players whether they are communicating with correct players (coming from different domains), or with an insider attacker who is equipped with local knowledge of remote domains. In other words, communicating players (e.g. human users, devices) from different domains, cannot easily trust each other.

1.2 Definition

Before discussing the issues further, some definitions that will be frequently used in this dissertation are given in this section, as some of these terms may be defined differently in other works.

Domain : unless a specific indication is given, by the term *domain* in this dissertation, we denote the scope of security policy rather than geographic location. It acts like a *security* domain.

Local/Localisation/Localised : by "local/localised/localisation" to a user, we mean being part of the user's domain, not geographic locality. For instance, a client Alice and the bank have different local domains.

Strangers : the definition of *strangers* is based upon the understanding of *domain*. Compared with a localised user, communicating principals (e.g.

human users, devices) will be considered as "strangers" from a user Alice's perspective, if they are not known before and have not established any prior security relationship (with Alice). Most likely, in this dissertation, strangers are usually from different domains to Alice. In this case, strangers' domains are described as *remote or foreign* domains for Alice.

Multiple domains and pervasive environments : the term multiple domains and pervasive environments are used *interchangeably* in this dissertation. It includes (but is not limited to) any pervasive computing applications that *genuinely* involve many different domains.

Confinement : in the multiple domain context, this term describes the problem of *confining* an attack from a domain within that domain (the analogy is similar to the confinement problem discussed in Lampson's paper [76]), particularly for the interactions between two distrusted users. Usually, when an insider attack occurs, confinement is needed to ensure that damage in one domain will not result in the damage in other domains.

1.3 New Challenges

With the rapid development of pervasive computing [104, 108, 128, 129], multiple domain applications are increasingly commonplace. Some of their major characteristics which challenge the conventional security paradigm are described below:

- Cross boundaries: in pervasive environments, an individual (security) domain often spans several organisational or geographical boundaries. It is important that the proposed security framework can support cross boundary management.

- Open surroundings and spontaneous interaction: pervasive users are going to interact with resources, or with each other, whenever and wherever

they would like to. Such a highly spontaneous nature brings *openness* to most pervasive interactions, for instance, access an on-line shop in a public cafe, or a peer-to-peer mobile ad-hoc file transfer. This requires the proposed security framework to deal with some previously *unknown* devices or human users, i.e. strangers.

- Transient associations: the associations among pervasive users are normally only for a (really) short period of time. For instance, Bob uses his PDA to withdraw cash from a ATM in a street that he has not been in before. The secure association between his PDA and this ATM will be terminating as soon as the completion of this cash withdrawal. Thus, the proposed security framework should be sufficiently flexible to support transient (cross domain) associations.

- High levels of decentralisation and heterogeneity: different domains have different assumptions, and implement different security systems. It is infeasible to place a globally unique infrastructure in practice. To do so essentially forces every domain to use the infrastructure's assumptions. The proposed security framework must give every domain the freedom to choose the detailed security implementation (for their own domain) according to their own security policy.

Above all, we can see that our target computing environment will be large-scale, highly decentralised, pervasive and more importantly span multiple domain boundaries. As a result, the security problems will become more complex and be infeasible to be addressed in a single step. A swiss army knife [2] security approach which takes protocols developed for the single domain case as attempts to apply them to the pervasive environment will be unlikely to successfully address the different threats imposed by different domains or different

[2]In paper [92], Needham pointed out that a swiss army knife protocol may not always be a good approach for security.

applications.

When interactions involve two untrusted users (or strangers) from different domains, an old paradigm to address this problem is to create a trusted environment by employing a global unique *trusted infrastructure*. A typical example is many Trusted Third Parties (TTP) based approaches, such as Public Key Infrastructures (PKIs). They let the infrastructure pre-define a set of assumptions that a trusted environment should meet. Thus, a security system or protocol implemented successfully in one domain can be migrated to different domains involved in the communications, combining with both important and necessary cross-domain authentication mechanisms. Explicitly, this is to invite and recruit many different domains into a big single domain. In a conventional network environment, the security resources and policy are statically configured within domain boundaries, no matter whether the network is inter-connected in a wired or wireless way. Thus, such a security understanding ("*big single domain*") has arguably addressed the security concern when many different domains are involved.

However, this is not a good idea even if we could do it. To have global trust is actually distracting us from what is more important in our own domains. In the multiple domain context, information and resources are provided by different organisations and other human users, even in a single physical environment. Interacting players from different domains are not likely to know each other, or have communicated before. It is not only infeasible to create a trusted environment to which those players can connect; more importantly, it is not what we want. We do not want the infrastructure to deter us from our own local policy, as far as security over multiple domains is concerned. Thus, a guarantee from a TTP may compromise security, and cross-domain authentication is not something that we should build on top of.

1.4 My Approach

In contrast, I would like to focus upon security over multiple domains as a starting point. I argue that security in a single domain is best seen as a special case of security over multiple domains. The richest case of security over multiple domains is the revolutionary pervasive computing environment. In pervasive environments, we are going to talk to *stranger* [3] users/domains frequently in a dynamic and spontaneous manner.

An interesting threat which emerges now is, *how to talk to the correct stranger*, particularly without the presence of a global fixed long-term trust infrastructure. I am motivated by the problem of talking to correct strangers in the context of pervasive environments from the beginning of this research. Explicitly, I would like to design a security countermeasure that is independent of any global universal trust infrastructures. More significantly, the approach has to be extremely efficient and easily implemented, considering heterogeneous single domains in the multiple domain context.

Thus, I introduce an important concept of *localising trust* design principle in security policy, which encourages domains (precisely, domain servers or administrative authorities) to get involved with their local users' communication. In essence, the *Localisation of Trust* (or LoT) security framework is the result of this principle. I consider security from the system architecture perspective, and system implementation is out of the scope of this dissertation. Thus, LoT is an architectural framework rather than the implementation of specific security systems. There are plenty of systems available in the market which can be used to implement the architecture. More importantly, I argue that the architectural issue is really important for the security in pervasive environments.

[3] Recall section 1.2, users or domains are said to be *strangers* if a secure knowledge (e.g. crypto-key information) or trust relationship has not been established previously among them.

The central part of the LoT framework is the novel dual capabilities model (or DuCaM) and encryption-based access control method. Moreover, a new form of certificate, Profile Certificate, is embedded into the dual capabilities. Profile certificates are used only for conveying information between players and their local domains, but not for making a final commit/abort decision for players. Instead, the ultimate security (commit/abort) decision is controlled by local domains, as inferred from the encryption-based access control. Eventually, a player in one domain can make a correct security decision without the struggle of understanding the precise semantics of security mechanisms in other domains.

1.5 Contribution

The primary contribution of this thesis is the design of a novel, localisable security framework, *Localisation of Trust*, or LoT, for multiple domains. LoT is designed to address many of the emerging research issues described previously. The essential concern is to convince readers that the LoT framework is an efficient and effective approach, when talking to *correct strangers* (under different assumptions) is desired.

Briefly, as a matter of principle, each domain in the LoT framework is encouraged to deal with their own security explicitly, rather than messing around with that of other domains. Conventionally, most security systems heavily rely upon foreign authorities' final commit/abort decision-making even when this was designed for a completely different purpose. In contrast to those systems, in the LoT framework the security decision-making for each particular transaction is ultimately controlled locally. A significant benefit of doing this is that users will not need to understand precise semantics of security mechanisms in other domains.

The key contributions are described below together with the chapters where

the relevant work can be found.

- Identifying the *incorrect strangers* problem in the multiple domain context, and proposing a design principle in security policy that guides the design of security countermeasures. A principle called *localising the trust* is presented, with the intention to serve as a general foundation for the design of future security systems, including (but not limited to) LoT. (Chapter 3)

- Designing a generalised model and basic infrastructure for the LoT framework. Another important part of this work is to design a tool-kit like set of security mechanisms, examining LoT from both authentication and access control ends. (Chapter 3)

- Proposing the concept of taking advantage of the positive human context to solve security issues in pervasive environments, and developing a human-centred authentication scheme with 2-channel approach to replace conventional (strong) ID-based authentication. (Chapter 4)

- Designing a new form of certificate, *profile certificate*, to support the concept of domain-oriented access control approach for multiple domains. Its purpose is only to convey (potentially incorrect, but probably correct) information between users and their associated domains. (Chapter 5)

- Designing the basic architecture for Encryption-Based Access Control (EBAC) to achieve domain-oriented approach with flexible delegation, and evaluating the advantages of revocation brought by EBAC. This is the essential model for the LoT framework, and a novel application of Identity-based Encryption techniques. (Chapter 5)

- Designing a Dual Capabilities Model, an extension of traditional capability-based access control models (e.g. I-CAP), and developing a detailed

protocol to fulfill the EBAC method. (Chapter 6)

The LoT framework does not intend to address the following challenges:

- Providing context-awareness services, for instance, (physical) location information. This results from the different aspect from which the pervasive computing environment is understood in this dissertation. As indicated above (see section 1.1), I focus upon the multiple domain context which appears in most pervasive applications rather than their "smartness".

- Designing specific security mechanisms for a single domain (or a local domain). This area has been considered by much past research work and is now considerably matured. Moreover, LoT intends to give every individual domain freedom to choose appropriate security mechanisms to fulfill their own security requirements. Thus, our purpose is rather to ensure that the heterogeneity challenge for the multiple domain context can be met.

- Supporting privacy. For some applications, confidential personal information can be gathered and provided by pervasive environments. Although privacy is also one of the new challenges for pervasive environments, the discussion on this is out of the scope of this research.

1.6 Dissertation Outline

In this dissertation, I will describe the research work (essential process towards to the design of the LoT framework) that has been done in past years. My description will be focused upon the problem of security over multiple domains with some example cases (which will be introduced gradually from chapter 4 to chapter 6) from pervasive environments.

The rest of this dissertation is organised as follows:

Chapter 2 provides a structured overview of some related work. Most of them have made significant direct impact on the outcome of my research, and the rest have provided indirect inspiration upon which this dissertation depends. This chapter begins with a review of the authentication problem in both conventional environments and pervasive environments, followed by descriptions of various authorisation (or access control) schemes. Then, it gives an overview of conventional revocation systems but understood from the delegation perspective. The chapter ends with a brief review of public key cryptography with a focus on identity-based cryptography.

Chapter 3 introduces the notion of *localisation of trust* and the overall infrastructure for LoT. The intention here is not only to form a solid foundation for the following chapters, but more importantly to explain why LoT is needed for security over multiple domains. The LoT framework is briefly described in a logical diagram, threat, policy and countermeasure, that follows the classic wisdom of designing a successful security system. Prior to this description, the paradigm shift in security for the multiple domain context is discussed in detail.

Chapter 4 describes a security mechanism for one end of LoT, authentication. This chapter includes a discussion of the essential unnecessity of having a *strong* ID-based authentication for pervasive environments. It also argues that the unique property of pervasive environments, human context, can be used positively to leverage the effectiveness of mechanisms achieving authentication. Finally, this chapter presents a two-channel authentication scheme and gives details of two protocols to demonstrate the usefulness of the scheme.

Chapter 5 illustrates a domain-oriented approach for the other end of LoT, authorisation. It provides descriptions of several concepts, including

profile certificates, remote authorisation, and localised authentication, that our domain-oriented access control mechanism builds on. Moreover, this chapter describes an architecture for Encryption-based access control (EBAC) that is used to achieve LoT's domain-oriented approach for access control over multiple domains.

Chapter 6 extends the concepts proposed in chapter 5, and describes an implementation (in the protocol level) of the domain-oriented EBAC approach using dual capabilities and partially trusted agents. The second part of this chapter discusses a guest access scenario that can easily be mis-regarded as a case of access control over multiple domains.

Chapter 7 concludes this dissertation by providing a critical evaluation of this research against the thesis goals described in chapter 1 (see section 1.4 and 1.5). Then, it summarises the main contributions, identifies various interesting features of LoT, and briefly discusses some potential future research work. This chapter also gives some examples of other areas that may benefit from deploying the techniques generated from this research.

Chapter 2

Background

In this chapter, we will present some previous works that are related to this research. However, it is not our intention at all to provide a general survey of the state-of-the-art of those related works. Essentially, we will review them regarding the problem that we are looking at in this dissertation. Thus, we would like to focus only upon the aspects affecting this work, and further to identify the *problems* of previous works. Note that, when the word *problem* is mentioned in the rest of this chapter, it is by no means for the purpose of criticism or negative implication. In fact, most of security systems shown in this chapter work perfectly in their own contexts and assumptions to solve the *problems* they are designed to solve. Nonetheless, those security systems/mechanisms cannot be simply copied to other contexts, although a lot of people do so in practice. In fact, it is also not those original system designers' intention to do this.

For most security systems, *authentication*, *authorisation*[1], *delegation* and *revocation* are the usual security mechanisms that are used to achieve desirable security services (i.e. confidentiality, integrity and availability). Thus, we will describe the conventional view of these major mechanisms, and start an argument why the underlying implicit assumptions behind them may not be

[1]Authorisation here refers to the final stage of access control. We shall return to this point later in section 2.2.

the right ones for pervasive environments, where the multiple domain concept in this dissertation is focused.

We will start by reviewing authentication from both conventional environments and the mobile context. Next, the concept of authorisation is described with the focus upon the capability-based distributed systems, followed by a review of delegation from the perspective of delegating both rights and responsibilities. We will then describe various schemes of revocation. Finally, the major relevant work on public key cryptography including Identity-based cryptography will be reviewed.

2.1 Authentication

In most literature, *authentication* is very often used to denote *identity authentication*, that is, to answer the question "who said this" [19, 74]. It is traditionally measured by one of three things, "something you know", "something you are" or "something you have", [109], depending upon the type of identification being used. In recent years, however, a broader understanding of authentication has emerged with the paradigm shift in the mobile context, i.e. mobile ad-hoc network and pervasive computing. Hence, we are going to review some works related to providing authentication in both conventional environments and the mobile context.

2.1.1 What Authentication Protocols Are

Authentication is conventionally an important process to guarantee that principals are who they *claim* to be [19, 82, 93]. It is normally accomplished by verification at the other end of the communication from the claimant. Thus, loosely speaking, an authentication protocol contains two major steps in the real-time implementation. One is to construct a *credential*, something you need to prove your identity explicitly. Most likely, a *digital identity* will be

represented in some form involving cryptographic keys [96]. The second step, known as *verification*, is to check the correctness of the credential provided by the claimant, to avoid impersonation.

- Authentication credentials: in real life, the credential used to prove ourselves can be many things, e.g. an ID-card or an officially signed document, or a set of pre-arranged secret codes or phrases. Similarly, in the digital world, the credential can be a shared key, a digital signature, or a capability[2]. Thus, when a client requests a service, such as file access, establishing a communication, and so forth, the client has to prove to the verifier in some way, that they are in possession of those credentials, and expects the recipient to recognise such credentials (which is the verification step) so that the request can be granted.

- Verification: in most cases, this means to verify credentials. Typically, verification is based upon sufficient knowledge which is frequently pre-obtained. For instance, if using a digital signature as the authentication credential, a verifier has to make sure that the correctness of the associated public key has been validated already (also that it is still valid when the request is committed). If a hash chain is employed to construct authentication credentials, the hash pre-image ought to be delivered to the verifier first (as shown in paper [5]).

For authentication protocols, it is important that the verifier is able to recognise those credentials correctly, particularly when the claimant and the verifier are in different domains. In conventional environments, it is usual to rely on

[2]The word *capability* here is different from the capability-based security systems that we are going to introduce later in section 2.2.3. The capability here is a more general term, and can be achieved by using certificates, one-way hash functions [56], hash chains [5] or observations from the communications in secret societies [17].

a unique trusted infrastructure implemented in every domain, when the authentication has to cross domain boundaries. Nevertheless, the nature of authentication is different when the communication is on the fly. The difference will be discussed in section 2.1.3.

2.1.2 Authentication in Conventional Environments

Authentication in conventional environments has been considerably well-developed for decades, from the early Needham-Schroeder authentication protocol [93] to public key certificates [65]. It always involves some form of proof of identity. For the purpose of simplicity, here, we categorise those typical (conventional) authentication techniques into two scenarios, *Storable scenario* and *Delegable scenario*.

- Storable authentication

  This scenario includes password, biometrics, and secret key systems such as Kerberos[3] [86]. Basically in these approaches, it is necessary to keep a centralised database of information based upon users' secrets (e.g. hash value of passwords, users' secret keys) and of personal biometric data. The decision of authentication is based on the results of required input information computationally matching pre-stored information in the centralised database. Thus, strong security assumptions for both challenge-response identification communication paths and remote servers are required. One problem of many storable authentication schemes (i.e. the password or biometrics applications) is the infeasibility to terminate authentication. For many Internet authentication systems for e-commerce, we need to register at the site with our *carefully chosen* username/password and input our credit card information for the authentication purpose. As argued in [110], there is no way to terminate

[3] In terms of the "ticket" element, Kerberos can also be considered under the second scenario.

the authentication if we no longer want an ongoing relationship with the e-commerce site. We cannot revoke our username/password because they never end (no expiration date associated with them). Also, the site will have our credit card information in their database, which we really do not want. In addition, the *confinement* problem emerges, particularly when users in one domain attempt to interact with other users from other domains. The compromise of one assumption (for example some attacks on poorly chosen passwords [58]), or one domain (for example secret user data in one domain is leaked) can bring about the collapse of trust in the entire architecture. Thus, the second scenario has been considered to address such problems.

- Delegable authentication

  Compared with the storable scenario, delegable authentication has earned much wider application due to its strong authentication performance with the assistance of public key cryptography. This approach includes public key certificates (as applied in PKIs [65] and PGP [132]), and tokenisation (or smart cards). Furthermore, we also lump capability-based systems [89, 69, 57] into this scenario just for now, due to their crucial relation between authentication and access control. Capability systems were originally oriented towards access control rather than authentication, to the point that many papers do not regard them as authentication mechanisms at all. We shall return to this point below (see section 2.2.3). For the delegable scenario, the output of the authentication process is justified by the input of authorised delegated capabilities being recognised. It scales the proof of capability (e.g. certificate chain) from the dedicated Trusted Third Parties (TTPs) or authorities in a transitive way. The "triangle" of trust, among prover, verifier and global TTPs in

one domain, is usually required during the authentication process. Consequently, the *Trust Transitivity* problem [16, 25] arises. If Alice from domain A trusts Bob from domain B on a particular event, and Bob trusts Claire from domain C, then, the meaning of trust transitivity is to allow us conclude that, Alice from domain A trusts Claire from domain C. Inevitably, this results in implementation difficulty (e.g. global TTPs topology) and uncontrolled imposition of trust (e.g. the unfair compulsion to fully trust arbitrary TTPs), particularly with the communication crossing different domains in a manner which may be hard for users to predict (which domains will a communication go through?).

In short, most conventional authentication schemes intend to construct a binding between human users' identities and their electronic identities, for instance, a binding "people → key → capability" described in [27] (and usually deployed in the delegable scenario, e.g. SPKI [43]). Such a binding is essentially a task of a local domain in the first place, e.g. issuing a credential to guarantee this kind of binding. Thus, when users attempt to interact with some other users outside his (or her) own local domain, many TTPs present, and are chained together to negotiate the recognition of, those credentials (e.g. certification path in the PKI). However, there is no guarantee that the success of security policy checking in one domain will be propagated across other domains. Moreover, it is infeasible to deploy such an approach to some highly decentralised applications, e.g. mobile ad-hoc communication.

2.1.3 Authentication On The Fly

Research for the security of mobile ad-hoc networks provides a different way to understand authentication, particularly with the development of many pervasive computing applications. Increasingly, authentication is required between two stranger entities (e.g. human users, devices) that are physically co-located but without prior trust relationships. Such a highly decentralised, mobile and

spontaneous mode of communications ensures the infeasibility of having a TTPs approaches (e.g. Kerberos) to achieve authentication.

Authentication in decentralised circumstances is frequently referred to the key exchange problem. This has been notably clarified by the Diffie-Hellman protocol [41]. Two communicating parties self-generate an identical fresh session key that is mathematically relevant to exchanged random bit-patterns between them. The preliminary of DH key agreement protocol is the assurance of integrity in the message exchange channel, otherwise *Man-in-the-Middle* attack will take place. However, such an assumption can barely be met in the mobile ad-hoc context when the wireless radio channel is used. Many protocols [28, 9, 8, 50] have been approved to thwart Man-in-the-Middle attack by manually involving a prior context (e.g. typing password or nonce value). These key exchange protocols provide a good way to consider authentication in circumstances that do not involve TTPs. Nevertheless, to exchange a prior context securely will be cumbersome, when interactions intend to move to many open and public locations. Peek-over-shoulder or relay attack can occur in a similar way identified in "Chip and Spin" [6].

Efforts to secure pervasive computing and fundamental communication structure (mobile ad-hoc networks) have received increasing attention in the security research field, and are described in many papers from different perspectives. Many researchers have raised the possibility to design additional security protocols which involve another physically independent secure channel (apart from the wireless radio channel). Some necessary cryptographic materials (e.g. sharing the same session key, the knowledge of public key certificates) can be transferred via this secondary channel, and be verified later on in the communications.

In their novel "Resurrecting Duckling" security model [120, 121], Stajano and Anderson mark a *physical contact* idea as the second channel to bootstrap trust between strangers. Moreover, based upon this approach, Balfanz *et al.*

[13] introduce a pre-authentication process to exchange some relevant cryptographic material in a demonstrative identification (physical recognition), authentic and secure location-limited channel (such as infrared, ultrasound or a short-wire). The image recognition from cameras [29], or a small token [31] can also be regarded as such a channel. Then, the cryptographic information (*e.g.* symmetric key, public key or other types of secrecy) exchanged via such a channel will be deployed to authenticate or verify the following communication in the open wireless medium.

In the spontaneous networking proposed by Feeney *et al.* [45], ad-hoc network users retrieve a session key via IR (infrared) channel and use it to secure the subsequent communications occurred in the insecure wireless RF (radio frequency) channel. Given a simple example, a project team coming from a number of organisations, gather together in a meeting room, the host uses a PDA to generate a session key (s) for this meeting and distributes it to each present meeting participant in a short-range IR communication among each PDAs. Consequently, the session key s is shared by every meeting participant and ready to encrypt the communication in the wireless radio channel.

A similar idea can be obtained from Capkun *et al.* [22]. The authors intend to use a short range connectivity mechanism, e.g. infrared or wire, as a secure channel to exchange some cryptographic material (a binding of user's name, public key and node address in cleartext or in the form of hash value) and then verify the signature or hash value of these prior-context submitted in the radio channel. Furthermore, they extend their approach by introducing a "friends" scenario to help establish a security association even between two strangers, employing the same philosophy. One person (i) can require the binding (name, public key and node address) of another person (j) from a "friend" f (who has already established communication with j) signed by f, vice versa. Potentially, they attempt to link the public keys with corresponding devices but avoid

hierarchical or chained trust transitivity from trusted authorities. Despite the situation that the authors consider central authority to be on-line only at the initialisation phase and kept off-line afterwards, the involvement of unique identity assigned by authority and (traditional) digital signature makes key freshness and revocation cumbersome.

2.1.4 Authentication

A twin-channel threat model for ubiquitous computing is also suggested in Creese *et al.*'s papers [34, 35] with the sampled protocols involving public key certificates. One channel is the communication medium channel with unreliable security, and the other is a more costly channel with higher security. Meanwhile, Wong and Stajano [130] propose multi-channel protocols between two wireless devices (at least one of which has camera ability) to perform authentication. They pinpoint the fact that a single protocol may get benefits from the use of different channels. The number and choice of channels that a protocol can employ should depend upon the nature of different applications.

Consequently, as far as authentication is concerned in the mobile context, we understand from the review of those works that its primary purpose is to bootstrap the trust between two strangers. When the communications are on the fly, such trust can be easily established from the deployment of additional channels, rather than from knowledge of strangers' identities. Moreover, it is considerably more convenient to have those secondary channels because mobile ad-hoc communications normally occur when the end points are in the vicinity of each other.

Thus, in this dissertation, the term *authentication* is enlarged to a more generic and broader term providing the authenticity of some necessary information, instead of conventional identity authentication. For instance, it may refer to authentication of a public key [4], or authentication of a credential. I

[4]From this perspective, I also regard the access-control oriented certificate-based systems

will analyse this point more fully in chapter 4.

The security requirements in conventional environments and mobile context are different in terms of authentication. Moreover, authentication needs to be regarded as an end in itself, because for many cases authentication on its own is the security requirement (e.g. bootstrapping the trust). However, more often, authentication serves as a foundation to achieve access control.

2.2 Authorisation

Access control is essentially used to prevent unauthorised users from reading, modifying or consuming information or resources, in addition to giving access permissions to authorised users. Conceptually, the access control systems firstly *authenticate* users using the appropriate authentication tools, then grant their access if the requests correspond to their access rights [5]. In this section, we mainly focus upon the final stage of access control, which is also very often described as *authorisation*.

Authorisation is usually regarded as the synonym of access control. The process of authorisation (the access permissions process) is managed by the *access control matrix*, that was introduced by Lampson [75] as a generalised concept to protect operating systems originally. An access control matrix contains rows that represent the active subjects, i.e. users in the system, and columns that represent the passive objects, i.e. information or resources of the system. In practice, thus, it appears in two separate forms, *Access Control List* (ACL) that views the matrix by columns, and *Capabilities* that view the matrix by rows. The access rights are specifically determined by an entry in both cases.

to achieve authentication as a goal (in section 2.1.2).

[5] Alternatively, it can be achieved by encrypting data in a way that only authorised recipients can decrypt it.

In this section, we will give a brief discussion of ACL and extended role-based access control. Then, we will focus upon capability-based access control.

2.2.1 Access Control List

A simple example of ACL is illustrated in figure 2.1. When a user requests

User	AccessRight
Alice	Read Only
Bob	Read, Write
Carol	Read, Write

Figure 2.1: *For each object, e.g. information data, resources, an ACL contains a list of users and their associated access rights.*

an access on a particular object, the system will look up the ACL for that object to check whether the user has the appropriate access rights. ACL is easy to implement because the access policy is managed centrally. However, as explained in [4], it is less suited for the environment when the user population is large and constantly changing, because every user needs an entry in the ACL.

2.2.2 Role-based Access Control

The idea of Role-based Access Control (RBAC) is familiar from the implementation of the user *groups* described in UNIX. When we look at the user pool more closely, we shall find that a large number of users can be categorised into a small number of *roles*. We can think of those roles like the job functions within a big organisation, e.g. directors, managers, secretaries or technicians. Thus, only *roles* are needed in a system, and the necessary *privileges* can be assigned to a *user* via those roles. Privileges here are a collection or a set of access rights that a role is permitted to carry.

Many different systems based upon RBAC [11, 53, 95, 106] have been proposed since the 90's. As illustrated in figure 2.2, however, the essential components in RBAC are users, roles and privileges. A RBAC system first assigns a user one (or more) pre-defined roles. Then, these roles will be checked to see whether the access request is under the privileges associated with those roles. For a single domain, RBAC has its own administrative advantage. For

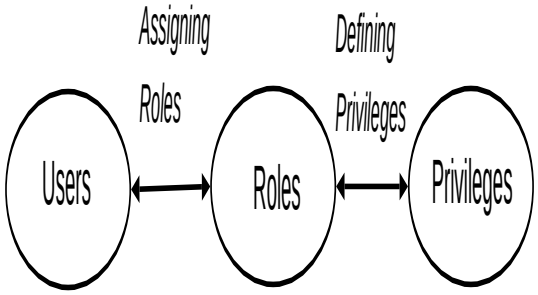

Figure 2.2: *RBAC systems map users to roles, and then roles to privileges.*

instance, when a user just joins the domain (or changes his job function), it is considerably easier to assign this user a role than directly assign access privileges to each object. However, it will be a struggle for a domain to understand a stranger's role (assigned by the other domains) in the multiple domain context.

2.2.3 Capability-based Access Control

Alternatively, the access control matrix can be stored by rows, which is called *capability*. As shown in figure 2.3, a *pure* capability is more flexible and can be passed from one user to another user, compared with ACLs.

However, the *pure* capability is vulnerable to forgery, copying or modification. Particularly, if a user possesses a capability, he/she is entitled to exercise the access rights that are associated with this capability. It is hard to check if the possession of this capability is genuine, or this user just simply steals or forges this capability.

User	*Access Right on Object1 (e.g. "Word Processing")*	*Access Right on Object2 (e.g. "Email client program")*
Alice	Read and Write	Read only

Figure 2.3: *For each user, a capability specifies what access rights are granted to the possessor of this capability on each object.*

Thus, the early Amoeba capability system described by Mullender [89] deploys a protection mechanism, a tamper resistant and trusted F-box, to address this problem [6]. The purpose is to transform a port name into a capability using the F-box at the network level. The port name is kept secret from the user due to the one-way function of the F-box. Thus, when the capability is presented by a user, Amoeba inputs the port name though the F-box and compares the output result with the user's capability. Moreover, the capability in Amoeba can be passed from one user to another user freely, and Amoeba does not regard this form of propagation as a threat. However, the free propagation makes it problematic to enforce the *confinement* property [60], as a capability is both necessary and sufficient to access resources within the system, for example, the capability may be passed to a malicious user.

As a consequence, a secure capability is often deployed to ensure *confinement*. It is normally achieved by two general methods.

1. One technique is to check the ID of the user (who presents the capability) at the time of using the capability, e.g. Karger's S-CAP [69] or Gong's I-CAP [57]. In S-CAP, a capability is necessary but not sufficient to gain access. The system checks the current access control policy, whenever a

[6]Mullender did not actually implement his conceptual model as described. F-box still uses ID in the implementation and consequently Amoeba does not enforce security policies as Mullender originally intended.

capability is used. For instance, the system will check the identity of the user (who presents the capability), and look up the ACL to determine whether or not to grant an access. This approach is not flexible and requires the entire system apply the same security policy. S-CAP, however, was not designed for multiple domain applications. It is infeasible to assume every domain will, or can, adopt the same security policy.

2. The propagation of capabilities can be restricted, for example by limiting the delegation of capabilities to other users [115]. Note that I-CAP also uses this second technique to address capability propagation between domains and the capability revocation problem by constructing a propagation tree. In I-CAP, when a user Alice wishes to delegate her rights on a resource to another user Bob, Alice needs to submit this request to the resource server S (who issued her the capability). The resource server would generate a proper (external) capability for Bob. More importantly, the server can draw a tree to monitor the propagation of the access rights associated with this capability.

In contrast to some capability-based systems introduced previously, Gong's I-CAP [55] is designed keeping multiple domains in mind. Potentially, the domain in an I-CAP system is not geography-based. To clear the concern of capabilities being stolen or lost, it introduces two forms capabilities, external and internal capabilities. The resource server generates a random number R_0 for a resource $Object$. The internal capability ($Object$,R_0) is only known by the server. Then, the servers computes R_1, which is the hashed value of the group (ID_{user},$Object$,$AccessControl$,R_0) for a local user in the system. The external bit-pattern (ID_{client},$Object$,$AccessControl$,R_1) is delegated to the corresponding user. As a result, different users hold different bit-pattern which corresponds to capabilities for the same object. Unfortunately, I-CAP suffers from the freshness problem because the value R_1 is put within the

(external) capability instead of being constructed at the time when the access is requested. Essentially, the I-CAP system requires a good underlying authentication mechanism to address the freshness problem.

Nonetheless, a sound idea of I-CAP is the definition of two forms of capabilities, external and internal representation. Only one internal capability need be generated for each object in the resource domain. For different domains, distinct external capabilities are computed for the same object based upon the internal capability, domain ID and corresponding access rights. The internal capability is always kept secret in the resource's local domain. The external capability is forced to have different bit patterns for different domains for the "same" capability. If the external capability is considered as the access credential between domains, the compromise of one domain does not help the attacker to gain any information of usable forms of external capabilities in other domains. The collusion of some misbehaving domains still cannot help the attacker. Moreover, by separating capabilities, I-CAP actually solves the *confinement* problem implicitly. Also, it can avoid the domino effect of security compromise to the other relative domains.

The logic of *delegation* has been considered in many authorisation systems, particularly the capability-based access control scenario. It is an important concept abstracted from access control, and we are going to review it separately in the next section.

2.3 Delegation

When a resource is shared across domain boundaries, the access requests from users are not always *local*. By using *delegation*, a user can allow some other users, possibly in another domain, to do some operations.

2.3.1 Certificate-based Delegation Schemes

With the intention to support authorisation in PKI, SPKI [43] associates users' public keys with their access rights rather than their identities. In principle, every entity (e.g. a server, a user) has its own public/private key pair. They can sign a certificate for any other users to whom they would like to transfer some rights. The signed *authorisation certificates* bind the proper rights with the owner. Hence, when the certificate is verified by the service owner or other users, the attributes embedded in the certificate are checked against the access request. Authorities (access rights) performing some actions are delegated flexibly among public keys. Based upon the SPKI certificates, Aura [10] introduces some concepts of *delegation certificates*. It also delegates access rights with signed certificate, but provides a useful discussion for certificate-based delegation.

Another mechanism that has been used to achieve delegation is proxies [94, 118]. A proxy [7](sometimes called token) contains the information identifying both its delegator and the delegatee. Thus, these schemes require a user to prove its identity. This requirement is often strongly linked to the deployment of public key certificates. For instance, a delegator signs a proxy with its private key and hands over to a delegatee. When this proxy is presented to a third party (i.e. the service provider), this third party needs to contact the authentication server to verify the proxy. SAProxies [81, 80] cleverly address this problem by letting the delegator bind his public key certificate (PKC) with the proxy token. A digest (only the signature part) of the delegator's PKC is included in the proxy instead of the delegator's ID or name. By doing so, the proxy becomes a Self Authenticating Proxy (SAProxy). It contains a delegatee's name, (the digest of) the delegator's PKC, the resource (or object) identifier, and delegated access rights (on this particular object), in addition

[7]The term "proxy" used in this context is not related to *proxy servers/services*.

to the expiry date. The delegator signs this proxy before handing over to the delegatee. When the delegatee intends to initiate an access request, he needs to include this proxy into his signed request. Since the PKC contains the delegtor's public key that can be used to verify the proxy itself, a verifier can check in his *local* area without retrieving any information from on-line authentication servers for verification. This feature seemingly makes SAProxies a desirable candidate to solve cross-domain authentication problem.

Nevertheless, like other certificate-based delegation schemes, inevitably, a common problem for SAProxies is the inherent *revocation* problem, when the access requests cross domain boundaries. If a user in the chain stops trusting something, it is hard to stop the proxy chain of trust. Thus, the system still requires a unique trusted infrastructure to be deployed in every domain that has been involved in the chain of delegation, to update revoked certificates or tokens. This is not what we really want for the case of multiple domains.

2.3.2 Delegation of Responsibility

Differing from the delegation of rights that have been discussed above, Crispo proposes a neat idea of *delegation of responsibility* in his Ph.D dissertation [36]. In his viewpoint, delegation is not merely defined as delegation of rights. He points out that it may be necessary not only to delegate rights but also the responsibilities associated with these rights for some applications (where the sharing of responsibility would result in accountability problems). When a user Alice delegates some rights, e.g. open the shop's safe while she is away, to another user Bob, it is important to distinguish whether the access ("opening the safe") is performed by Alice, or by Bob acting on her behalf. This is where delegation of responsibility comes to play.

Delegation of responsibility may or may not imply that the associated rights are transferred from Alice to Bob. However, in this example Alice should not be able to open the shop's safe any more, while she is away. This

essentially reduces the amount of trust that is required to enhance the security policies.

Both delegation of rights and delegation of responsibility show the importance of *trust* between delegators and delegatees. The basic principle in delegation is delegators have to *trust* delegatees who will act on their behalves. We will look at this *trust* issue next.

As a concept, the notion of *trust* has been defined differently depending upon the context of applications. It is important to understand *trust* not only because many security systems are actually constructed from it (implicitly or explicitly), but more importantly because of its relationship to the need for revocation. A typical example is the trust management approach (see section 2.3.3), which is essentially delegated access control with the extended relation to *trust*.

In this dissertation, we are going to follow a simple definition of *trust* from Christianson and Harbison [25] to explain the principle of trust. Note that it only focuses upon some necessary principles of trust in the context of this dissertation. We do not by any means intend to discuss a concrete trust management infrastructure or trust model here.

In their work [25], Christianson and Harbison argue that trust can be considered as the conjunction of *Honesty* and *Competence*. Thus, "A trusts B on a statement of some knowledge X (in short, A *trusts* B)", can be unpacked as:

$$\text{If } (A \text{ believes } (B \text{ says } X)), \text{ then,}$$
$$A \text{ believes } X.$$

This statement can be further *translated* to the conjunction of,

- *trust in honesty*:

$$\text{If } (A \text{ believes } (B \text{ says } X)), \text{ then,}$$
$$A \text{ believes } (B \text{ believes } X).$$

- *trust in competence*:

$$\text{If } (A \text{ believes } (B \text{ believes } X)), \text{ then,}$$
$$A \text{ believes } X.$$

Thus, the belief generated from *trust* has to come from both *honesty* and *competency* statements. One statement alone cannot produce the belief. For instance, if *only* "A believes (B believes X)" is stated, it does not necessarily mean that "A believes X". Alice and Bob may have different security requirements based upon different threat models. This results in the fact that Alice and Bob may have different beliefs even for the same thing, e.g. Bob believes this program is running safely in his computer because of his faith in his updated anti-virus software. Alice does not know this. Alice may not trust the anti-virus software. Or simply, she may not have the anti-virus software installed at all. Moreover, neither of those two statements is reversible. If "A believes (B believes X)" is stated, it does not necessarily mean "A believes (B says X)".

Roughly speaking, a *belief* (on something) can be generated from either *knowledge* or *trust*. Knowledge (in the sense of true justified belief) is in principle transferable: we can pass knowledge over to other people [8]. However, in practice transferring knowledge is problematic in a distributed environment. Consequently, in protocol analysis, trust is often used to provide a *substitute* for knowledge (for instance, to justify taking the next step in the protocol). Statements of trust are not transitive, even in principle: Bob cannot transfer his trust in Carol to Alice. In other words, if Alice trusts Bob about something and Bob trusts Carol about the same thing, it does not necessarily mean that Alice will trust Carol about the same thing. Moreover, trust is non-monotonic

[8] In real life, the definition of *knowledge* can be more complicated than that which I have described here. Some "knowledge" may be abstracted from some "trust" statements. For those cases, *knowledge* cannot be passed on (similar to *trust*). However if A knows that B knows X, then A knows X.

which explains the importance of revocation (see section 2.4).

2.3.3 Trust and Policy Based Delegation Schemes

The root of trust and policy based delegation can be traced back to the idea of a *trust management* scheme. *Trust management* is first introduced by Blaze et al. in their PolicyMaker [16] and KeyNote system [15]. It provides a framework for specifying and interpreting users' security policies, security credentials and their trust relationships.

Policies in trust management systems are purely for *local use*, and credentials are usually signed certificates. These certificates are similar to SPKI, binding public keys to assertions delegating authorisation to perform actions (that they are trusted to sign for). A programmable language that is easy for humans to read is usually implemented in trust management systems. In practice, the resource owner (S) obtains certificates from users who request actions. After verifying the signatures on certificates as well as the validity of the certificates, S submits a user request, certificates and local policy description (of this particular action) to the local trust management engine (e.g. PolicyMaker). Then, the access decision will be made if approved.

From our perspective, a fruitful point is what Blaze called "locality of control" [16]. By supporting local control of trust relationships, a globally known hierarchy of certificate authorities can be avoided. This useful remark is being extended by many works [20, 38, 48, 62, 67, 114, 131] to construct a trust-based or policy-based system (mostly associated with the RBAC mechanism) in different contexts. For instance, Trust Establishment (TE) from Herzberg et al. [62] is designed to assign roles to strangers. The Trust Policy Language (TPL) is presented to map stranger users to predefined roles, based upon certificates that are issued by a third party (e.g. a TTP or a user). A TPL rule defines some requirements, such as the issuer needs to belong to a specific group, and conditions for joining a role. In his Ph.D dissertation [38], Das

Chowdhury looks at trust management from a different aspect and applies it in the context of *privacy*. His approach separates ID management from trust management and therefore delegation can be done *anonymously*. This is achieved by the use of a *surrogate*, which is constructed from the *ring signature* scheme [102]. A surrogate does not unveil a delegatee's ID but implies the delegatee is trusted by a delegator for this particular transaction. Thus, trust is localised within the system.

Kagal et al. [67, 68] target the security challenges which have arisen from lack of central control and rarely predetermined users in the pervasive computing environment. Their systems are also built from XML language to form *distributed trust* rather than just user authentication and access control. It is impossible for a *security manager* to understand a foreign user Bob's role in accessing some resources. Hence, Bob requests permission from a local user Alice, who can delegate access rights to anyone she trusts, according to local security policy. Alice hands over to Bob a signed delegation certificate that implies the proper access rights are delegated to Bob. Then, Bob's request will be granted after he sends the security manager this delegation certificate along with his identity certificate. The trust-based systems will make the final decision based upon the policy check, e.g. whether Alice's rights are revoked. Also, this final decision-making process can be based upon recommendations (as described in [114]) or history (see paper [48] [9]).

These trust-based schemes, do not themselves implement certificate distribution or revocation services. Instead, they have to rely upon some external programs because signed certificates for delegation are frequently used.

An urgent problem for most delegation schemes is how to revoke the delegated tasks, particularly for the chain of delegation. Again, when delegation occurs, the system needs to consider stopping the chain of trust once a user

[9]Following the social impact of *the small world phenomenon* [85], Galice et al. argue that two users can find the common users they have met before, when they meet for the first time. Those history-based elements are seen as a bond to manage trust.

stops trusting something. Hence, from our perspective, delegation has to be considered closely with *revocation*.

2.4 Revocation

The term *revocation* has frequently been limited to *certificate revocation*. In essence, an issued public key certificate (for identification or access control) has to be invalidated before its defined expiry date, perhaps due to a compromise of a user's private key, affiliation change, withdrawal of operations and other unpredictable reasons [47, 65, 90]. However, it is also sometimes necessary to revoke some specific delegated privileges, for instance, a capability, or a role in most access control systems. In this dissertation, consequently, by means of revocation, I mean the revocation of access in addition to (conventional) certificates revocation.

2.4.1 The Principle of Trust

Following the notion of trust discussed on page 41, there are three essential principles for *trust* as far as this dissertation is concerned,

- Trust is *subjective*: for the same statement, different users will have different viewpoints of trust based upon their own assumptions and threat model. It is infeasible to have global trust.

- Trust is *context-dependent*: the statement of trust is dependent on the specific context of applications. Interestingly, we can say, for example, "I may trust A to introduce a car sales assistant to me, however, I may not trust A to introduce a mortgage representative to me".

- Trust is *non-monotonic*: the statement of trust may (and most likely will) change dynamically over the time. For instance, "I trust A at time t_1", however, at time t_2 (where $t_2 > t_1$), I may not trust A, and I may trust A again at time t_3 (where $t_3 > t_2$). Thus, trust is not permanent.

Thus, some trust assumptions may no longer exist under some circumstances, particularly when users cannot tell the future. That is where revocation is needed.

2.4.2 Various Revocation Schemes

Consider (public key) certificates, it is revocation which makes certificates non-monotonic [79]. Conventionally, certificate revocation policies are usually decided by the CAs that issue the certificates. A typical implementation is the periodically published off-line Certificate Revocation Lists (CRLs) [65], either containing a *positive* (listing certificates which are still validated at the time of issue) or *negative* (the certificates which are expired or revoked) statement. Alternatively, on-line revocation authorities, which were introduced by Crispo and Lomas [37], are used in OCSP [91] to respond to users' queries regarding the validity of certificates. However, these approaches are not suitable for decentralised multiple domain environments. In fact, even in the conventional environment, significant expense is required to solve the revocation problem by using these conventional mechanisms [14], such as the transmission costs and the infrastructural costs.

A much more promising revocation mechanism is designed by Micali in his Novomodo system [83, 84]. Micali's revocation system involves a CA, one or more (publicly *semi-trusted*) directories. The basic scheme is based upon the use of hashed chains. For each user, the CA chooses a random number X_0, and repeatedly calculates its hashed value n times by using a public one-way hash function $H\{\}$,

$$X_n, \text{ where } X_i = H\{X_{i-1}\}$$

The value of n is subject to the revocation policies defined by the CA. For example, it can be total days of the month (e.g. "30"), if the CA checks the validity of certificates on a daily basis. X_n is included in the (traditional)

certificate along with other information. On the ith day after issuing the certificate, the CA sends $H\{X_{n-i}\}$ to the directories if the user's certificate is still valid. Otherwise, it does not send this value. Thus, when a user Bob submits his certificate to a resource server Carol on the ith day, Carol retrieves X_n from Bob's certificate, then queries a directory for obtaining $H\{X_{n-i}\}$. The verification is done once Carol finds whether the following equation stands,

$$H^i\{X_{n-i}\} = X_n,$$

where $H^i\{\}$ is to hash the value repeatedly i times.

If it is held, Bob's certificate is still validated, otherwise, the certificate is revoked (at least for that day). It is an efficient revocation approach, because,

- The verifiers do not need to trust the directories, as only CAs can produce the hashed pre-image of X_n. Thus, those directories are *semi-trusted*.

- CAs do not need to be on-line all the time. More importantly, Micali's revocation system can revoke users' certificates *selectively*. If a user's certificate is only invalidated temporarily, the CA can just stop sending the value of $H\{X_{n-i}\}$ for the "frozen" duration. Once a user is back into action, the CA can simply start to generate the proper $H\{X_{n-i}\}$ from this instance, without re-setting the system from the beginning.

Gentry [51] also adopts this neat idea but implements it using pair-based cryptography to address the third-party queries problem, when users use their public key pairs for encryption/decryption rather than signature.

Moreover, Rivest [98] suggests that the revocation policies should be set by the "acceptor" (for instance, the service provider or a third party verifier) rather than CAs. The "signer" (i.e. the certificate holders) should supply the necessary *recent information* (with the assistance from CAs, e.g., short-lived certificates or some recent statements as described in [123]) as the evidence

of its validity instead of another way around. He also argues that revocation should have different semantics with respect to different reasons to revoke. For example, for (private) key compromise, it is unnecessary to get CAs involved at all. The certificate-holder can simply sign a *suicide note* declaring the key has been compromised or dead and publish it. The idea of *suicide* is also used by Moore et al. [30, 88] recently to address node revocation problems in ad-hoc networks [10].

Revocation has always been viewed as a both difficult and complex feature to implement, particularly in a capability-based system [70]. It is hard to grasp, although some previous works have examined it from different perspectives. The revocation of access privileges has been discussed by Khurana and Gligor in [73] (using attribute certificates as the example). They propose that it is necessary to do *selective revocation*, when both attribute certificate and identity certificate are implemented in the system. Meanwhile, *transitive revocation* is needed to revoke delegation chains.

One interesting way to fulfill revocation in the *context of confidentiality* can be observed from ID-based Cryptography (IBC) which will be reviewed in the next section.

2.5 A Short History of Public Key Cryptography

In this section, we will give an (over-brief) history of public key cryptography (PKC) with the attention on IBC, which is a twist of PKC. Additionally, following the development track of PKC, we have also observed that many PKC systems (unfortunately) produce another problem to replace the one which they successfully resolve.

[10] Basically, when an ad-hoc networking node believes or perceives another (neighbouring) node has misbehaved, their "radical" strategy is to let this node publish a suicide note announcing both the misbehaving node and himself are dead.

It seemed that all security problems go away when public key cryptography appeared in 1976. In the PKC scenario, e.g. RSA [101], ElGamal [42] system and other variants including Schnorr's scheme [111], a user Alice keeps her private key secret as well as "broadcasting" associated public key to the others. It is (mathematically and computationally) hard to retrieve private keys based upon the knowledge of corresponding public keys. As a consequence, the other users are "guaranteed" that the message encrypted under Alice's public key can only be seen by Alice herself. In addition, any messages that can be verified by Alice's public key are "truly" signed by Alice. However, the great concern is the problem of *public key distribution*, that is, how to associate a public key with the correct participant as discussed by Christianson and Malcolm in [27], particularly in a multiple domain case. As illustrated in figure 2.4, it is not convincing for Bob from domain B that a public key claimed by "Alice" (from domain A) is really the one associated with Alice, not another public key "owned" by Evil.

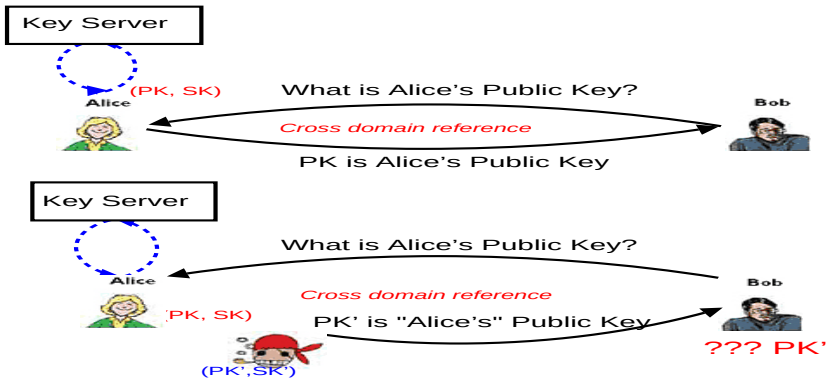

Figure 2.4: *Evil can interfere with the public key distribution channel and simply do "man-in-the-middle" attack.*

As a consequence, PKIs have introduced dedicated (trusted) authorities. These reside in each domain, and their sole role is to issue a fresh certificate associating public keys with the legitimate participants (mostly in the form of

their identities). Inevitably, trust transitivity [25] (from the chain of certificates) has to be required when the communication is taking place over multiple domains. This solves the public key distribution problem after involving an underlying trusted infrastructure. However, as shown in figure 2.5, it becomes problematic to revoke certificates. Moreover, a unique trusted infrastructure spanning every domain require every player to understand a lot about foreign domains.

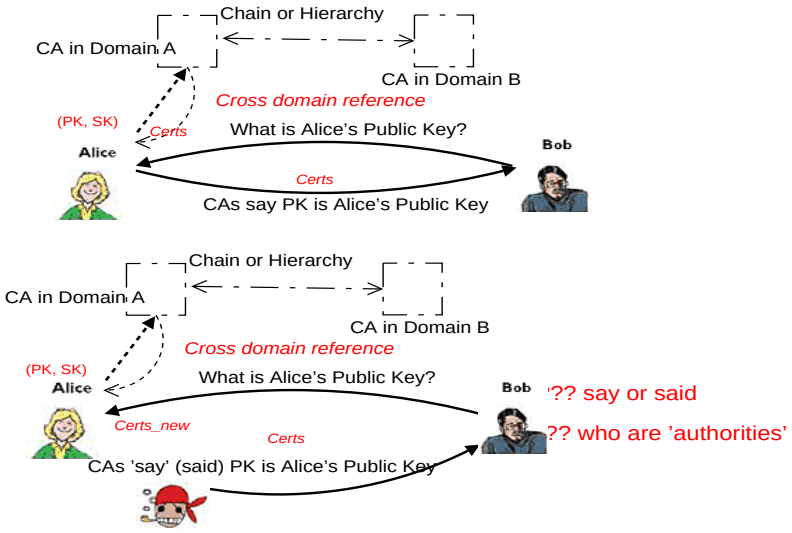

Figure 2.5: *Bob believes PK is Alice's public key only because authorities say so.*

The PGP scheme [132] provides a different way to target the public key distribution problem. It is a more decentralised approach than a typical PKI system. In practice, a PGP user can submit her public key to some well-known public keys repositories on the Internet. Then, she may employ another different channel over which she has most control (e.g. her personal webpage), to publish the fingerprint (hashed value of the public key) of her current public key. Thus, the third party can verify the authenticity of her public key obtained from a public repository. However, PGP does not really solve some inherent problems of PKC (such as certificate revocation requiring transmission of trust along key chains).

In 1985, Shamir proposed a different public-key cryptosystem, Identity-based cryptography [113]. IBC had not received enough attention until some schemes based upon pairing [18, 24, 52, 64] were proposed after 2000. Shamir's novel approach is for any pair of users to communicate with each other in a secure way without cryptographic key exchange, key repositories or directories, or services from a third party in a remote domain. In principle (as shown in figure 2.6), IBC associates a user's identification with key pairs by defining public keys as the ID (with some redundancies).

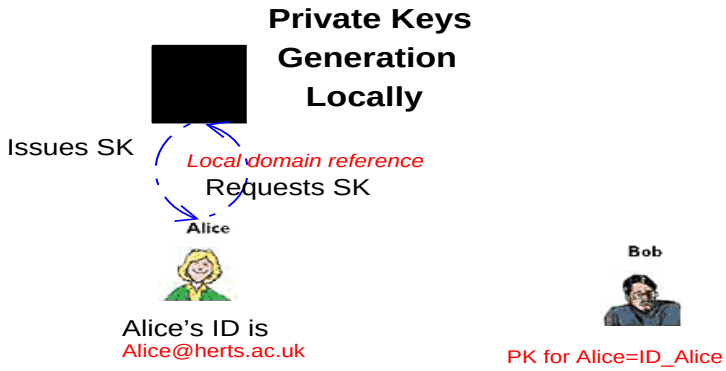

Figure 2.6: *Bob does not require any prior crypto-knowledge about Alice to construct her public key. Moreover, public key freshness is met by putting in some extra information, e.g.,* $ID_{Alice} = h(Alice@herts.ac.uk, currentdate)$

Based upon the description in paper [18], some cryptographic requirements for IBC are (briefly) highlighted below,

- Each domain has a local and (only) locally trusted third party, Private Key Generator (PKG). It sets up publicly known system parameters ($Params$) including the domain's public key, and a privately master secret, s (randomly chosen). More importantly, the PKG is also responsible for extracting a private key K^-_{Alice} for Alice. It is associated with her public key, $K^+_{Alice} = \text{IDinfo}_{Alice}$ by inputting $Params$, s and IDinfo_{Alice}. For the underlying algorithm,

  - Given a seed s, it is easy to compute a K^- for any possible K^+.

– Given any pair of (K^+,K^-), it is infeasible to compute s.

- When Bob needs to encrypt a message (M) using IDinfo$_{Alice}$, he does, Inputting $Params$, IDinfo$_{Alice}$ and $M \rightarrow Ciphertext(C)$

In IBC systems, a user, Alice, only needs to trust her local PKG for issuing her the right private key. From our perspective, this feature is really important for the multiple-domain context because trust is established *locally*. An additional authentication mechanism is required to be in place before the private key issuing process, however, it is not difficult to achieve this in a local domain. It is also a flexible approach, as the PKG does not need to generate a private key for local users beforehand. A player, Alice, can choose any arbitrary bit-patten as her ID regarding her local domain's policies. The PKG will be happy to issue her the private key associated with the *IDinfo* she submits, as long as she can pass the local authentication process.

In recent years, many other works have also been developed based upon the basic principle of IBC, including certificate-based [51] and certificateless [2] schemes targeting the inherent key escrow problem in the basic IBC setting [11], authenticated key agreement protocols [12, 116] and access control oriented schemes [61, 117].

A problem for IBC is that a malicious user Alice can set up a *fake domain*. Like other domains, this fake domain's public key is known to all players. Bob would expect Alice's domain to have a control over her misbehaviour by not issuing her the proper private key. However, since the domain is fake, Alice's private key is always available to her. IBC does not have an extra mechanism to stop this happening. However, the importance of IBC for our work is the mechanism of *localising the trust*.

[11] A basic IBC system allows the PKG to generate a user's private key and every user's private key is known by the PKG.

2.6 Conclusions

This chapter has presented an overview of the research on which this thesis is built, from authentication, authorisation, delegation and revocation aspects. It has also briefly reviewed the development of public key cryptography. We opt to describe only the semantics of those works rather than illustrating protocols in detail. Also, some of those researches are essentially overlapping and provide different security services.

In the succeeding chapters we will show how some of these works can be adapted for the design of our work in the multiple domain context. The next chapter will describe the basic infrastructure of our work based upon the nature of relationships, threat modeling, security policy and security countermeasures, for secure systems design.

Chapter 3

The Need for Localisation-of-Trust

This chapter introduces Localisation-of-Trust (or LoT), a security framework to provide necessary services in the multiple domain context. It begins with a description of the contextual change for security in pervasive environments. Then, in section 3.2 I illustrate an overview of the LoT framework, pinpointing the basic infrastructure and key concepts. These concepts serve as a basis for LoT. Section 3.3 explains a major issue concerned with LoT, talking to *correct strangers*. In section 3.4, the *localising of trust* security principle is described and the need of it for the multiple domain context is discussed. This chapter ends with the outline of the exampled security mechanisms provided by LoT. LoT examines them from two ends, authentication and authorisation, and proposes a *tool-kit* of security countermeasures for pervasive environments.

3.1 Paradigm Shift in Security

Security in the multiple domain context has not been highly researched as a whole in *conventional environments* (i.e. distributed computing, mobile computing [1]). A major argument is that a successful protocol targeting one domain can "easily" be lifted to multiple domains. This is mainly fulfilled with the

[1] Arguably, the mobile computing environment is not usually considered to be "conventional". However, from the *domain* perspective described in this dissertation, most mobile

necessary assistance from a globally trusted infrastructure. In essence, the same protocols are *unconsciously* being used for different context. A Swiss Army Knife like protocol has already been viewed with alarm [92]. It will be more dangerous to implement it in pervasive environments in particular because pervasive users are more likely from different domains that have different assumptions and threats. Moreover, many pervasive interactions may occur in several different environments. However, with the development of pervasive environments, the paradigm of security for multiple domains has shifted.

The most obvious change for pervasive environments is the basic wireless connection, frequently in the ad-hoc manner. If a pervasive environment only replaces wires with wireless RF media, then to secure pervasive environments is not too hard, considering the well developed cryptosystems described in chapter 2. However, the new challenges [35, 45, 121], for instance, poorly defined network boundaries, dynamic enrollment, no pre-configuration, transient association and decentralised infrastructure, have entailed a massive qualitative change in security requirements.

Most of all, as far as this dissertation is concerned, the primary change in pervasive environments is the need to provide necessary security requirements over multiple domains. Again, by means of *domains*, I emphasise the difference of security policies rather than geographic locations. Those requirements are different from what we have experienced in conventional environments (i.e. distributed computing or mobile computing). It is mainly manifest in the following respects:

- Prior knowledge for interactions. In conventional environments, we have pre-obtained knowledge about the communication which is going to occur. Hence, most security protocols heavily involve the necessary process of identifying communicating entities by their names (or their attributes)

computing applications are still based on access in a single domain, which could be considered a conventional scenario.

directly, or some ID-proof forms indirectly. The information used to prove identities explicitly could be, e.g. a password, a shared secret key or a digital signature. In a pervasive environment, we are most unlikely to know in advance the communication we are going to interact. Thus, it is cumbersome to retrieve those identity related proofs before interactions occur, or even to find out how to do it.

- Filtering bad guys in the network protocol level. Conventionally, different networks are physically distinguished and therefore network boundaries are clearly defined. It is feasible to place a properly configured gateway between two networks, analysing the incoming/outcoming data streams throughout the network. Any suspicious or unexpected information from outside of the network will be blocked if necessary. Most pervasive applications take place in a unpredictable way and are managed without any external configuration. Explicitly, such a self-organised network infrastructure implies the absence of sharp network boundaries. It results in the fact that the inside and outside of the network are not clearly distinguishable. As a consequence, a conventional gateway/firewall approach to secure the network protocol level cannot secure a pervasive environment, because the network boundary is not clear and well-defined [45]. In addition, to block unwanted traffic cannot address an internal attack. A malicious user in a pervasive environment can roam among many networks so that the attacker may already be inside network gateways.

- Dependence on a pre-established or centralised trust infrastructure. A conventional approach is to have some trusted third parties (TTPs) on a global scale, for instance, symmetric-key based Kerberos systems [86], early X.509 Public Key Infrastructure (PKI) [65] and SPKI/SDSI system [99, 43]. The security decision-making for a certain purpose is handled

by the presence of those TTPs, and we hope that they will be properly configured within a domain. This does not suit a pervasive environment due to the decentralised character of pervasive communications. More significantly, neither TTPs, e.g. Certificate Authorities (CAs), nor a global infrastructure is always accessible when a particular security decision has to be made on a specific communication session. This leads to the problem of delegation and revocation as stated early in chapter 2. Also, it seems to be unrealistic to require every pervasive user to carry their own infrastructures around, because of the computational resource limitations on those daily objects.

Considering these security impacts, the multiple domain context (in particular, a pervasive environment), requires a contextual change of nature for security. We must guarantee that we can identify the correct user and its properly associated access rights in order to achieve desirable levels of security. These security requirements are usually achieved by two ends, authentication [2] and authorisation. With the paradigm shift in the multiple domain context, those two security ends are changing when we intend to routinely talk to significant numbers of *stranger* human users or devices spontaneously and dynamically. We have no prior knowledge about the names/IDs, or roles of those to whom we are going to talk, or which kind of privilege a user needs to access resources. In the meanwhile, we cannot totally rely upon the infrastructure to make the right decision for us. In order to address those *classic security problems* but in a new context, I propose a Localisation-of-Trust (or LoT) framework.

3.2 Overview of the LoT Framework

LoT is a framework to provide necessary security requirements involving multiple domains, a pervasive environment for instance. It can also be used to

[2] In this dissertation, the term *authentication* is mainly used for user authentication if no specific meaning is given.

advantage in conventional environments (i.e. distributed system, mobile computing). However, the focal interest of my work is pervasive environments, as pervasive computing is the richest case of security over multiple domains. The LoT framework intends to investigate the security problems over multiple domains from the two logical ends, authentication and authorisation, due to the inevitable difference with the single domain case for the nature of bootstrapping trust. In a conventional meaning, authentication is to determine "who is speaking?" or, "who is making statements?" and authorisation is usually about "who is trusted?" or, "the statements made by whom are trusted" [1]. Generally speaking, authentication is mainly used to bootstrap access control. In this dissertation, by means of *authentication*, I mean this generic meaning of bootstrapping access control. I do not (just) mean conventional ID-based authentication. Thus, access control is essential and authentication is not primary.

In the rest of this dissertation, it will be noticed that most statements in LoT are heavily towards access control, or trust establishment. This is because, firstly, bootstrapping trust naturally requires something different; secondly, it is a harder problem to establish trust with variations. Last, authentication (the generic term) is essentially the minimal level of access control. Consequently, I focus more upon the access control problem, and the LoT framework will be "deliberately" presented in a more access-control-like style and expressions for the discussion.

The LoT framework is delivered as a result of acknowledging the significance of the local domain's knowledge. It is based upon a systematic policy of *localising trust*. As matter of principle, each domain in the LoT framework intends to deal only with their own security policy explicitly, rather than messing around with that of other domains. Most currently existing security systems tend to force a user to understand other domains' policies when he/she intends to access some resources in those domains. From security's viewpoint,

it is difficult and, I will argue, unnecessary to do so. Therefore, I would like to provide an alternative in this dissertation and propose the LoT framework. LoT is based upon the *localising the trust* security design principle, which encourages domains (precisely, domain servers or administrative authorities) to get involved with their local users' communication. In LoT,

- A user may be a human, a computer, a portable computerised device, a program, or a process, etc.
- A domain is very flexible depending upon the context and its own policy. A user can set up one domain or several domains for different security purposes and fitting into different environments.

Figure 3.1 gives an overview for the LoT framework. This infrastructure is designed to minimise trust assumptions, and provide desirable security services in an effective and efficient way.

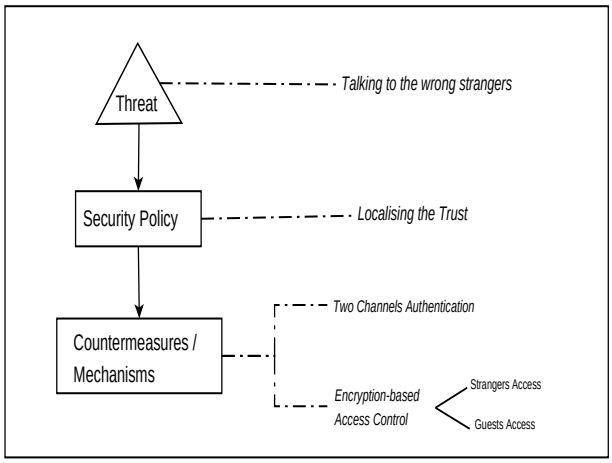

Figure 3.1: *The LoT framework overview*

1. **Threat**: In pervasive environments, we intend to talk to *strange users* frequently in a dynamic and spontaneous manner. Generally speaking, the main threat for pervasive environments is to talk to *incorrect* strangers. Those strangers are usually from many different domains. They may be a strange user from a friendly domain, or a friendly

user from a strange domain. Amongst them, most likely, secure pre-knowledge (e.g. crypto-key information) or trust relationships have not been established. For solving this, conventionally, an infrastructure, such as Kerberos [122], PKI, etc. is introduced to help users to determine correct strangers usually by verifying their IDs, names, long-term public keys, or roles. The trusted infrastructure has its required assumptions. Thus, inevitably, users have to set up their own assumptions and security policy in accordance with those of the infrastructure. However, the semantics of the threat model will be different due to the nature of applications, particularly in pervasive environments. The threat is specific to the requirements from different pervasive applications and assumptions which will not be known to the infrastructure. As a consequence, it must not let infrastructure decide local policy.

2. **Security Policy**: the main security design guideline in security policy is *localising trust* principle. It is difficult to establish *trust* in a pervasive environment, if users have to struggle to understand a foreign domain's policy and precise semantics of security mechanisms. Thus, we have to seek significant assistance from external TTPs. However, it becomes a harder problem as soon as a user has to trust any *arbitrary* external authorities. In LoT, by localising the trust, a user is able to put trusting things in his/her own domain, or some places that the user already has stable connection with.

3. **Countermeasures/Mechanisms**: Unlike most security frameworks that usually propose a static set of countermeasures, arguably, LoT employs a *tool-kit*, which contains several countermeasures for different contexts. I suggest three main examples that are used to target the threats and contexts considered in this dissertation. Consider the unique property of pervasive environments, namely, positive human context, that is,

human users have clear intention about what they are doing. I reason that it would be extremely efficient and effective, if the security framework allows users to choose a suitable mechanism from a tool set with respect to their own assumptions and the changing environments that they are facing.

3.3 The *Correct Strangers* Problem

To understand the main threat concerned in this dissertation, let us look at a more generic case first. In figure 3.2, a local player (let us say Carol) can access some resources in her own domain in two ways. Carol may locate herself inside her domain boundary. Or, she may locate outside of her own domain (usually in another remote domain). This is OK for one single domain, because a security manager is present to make sure all local users are clear about the local policy. Unfortunately, it is a different story in the multiple

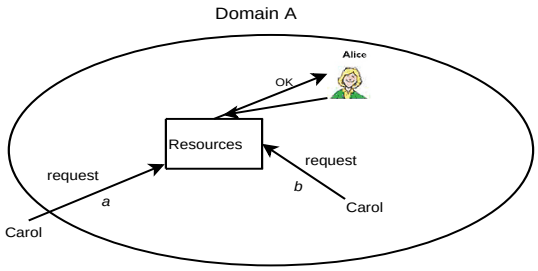

Figure 3.2: *Assume that Alice is the domain server for domain A. A local player Carol can use a pre-defined userID, a password, or a shared key proof to log in, and further access some resources in her own domain A.*

domain case due to the lack of knowledge and reliable resources about other domains. Thus, a domain's server has *no* local knowledge of players from other domains, as illustrated in figure 3.3. Users or domains are said to be *strangers* if prior security knowledge (e.g. crypto-key information) or trust relationships have not been established among them. Users are most likely to be *strangers* if they are from remote (friendly or strange) domains. A common resolution

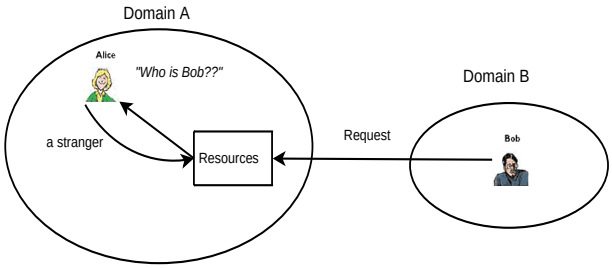

Figure 3.3: *Alice is again assumed to be the domain server for domain A. Bob is from a different domain B. Since Bob is unknown to domain A, Alice will regard Bob as a stranger. The request will be refused.*

adapted from conventional environments is to lift a security system/protocol implemented successfully in one domain to the multiple domain case. Two techniques [3] have been typically deployed to solve this *strangers* problem when communications involve different domains.

1. "Let you become one of us" approach. Alice treats Bob as a local user. She gives Bob the local user's privileges by setting up a new log-in account or assigning a role to him.

2. "Let you become my apprentice" approach. This is usually known as the delegation scheme. Alice delegates some of her rights to Bob, and authorises him to access some certain resources on her behalf but with some necessary restrictions.

For accomplishing these approaches, a universal global trust infrastructure is needed, as simply highlighted in figure 3.4. It is doubtful whether this can be achieved in pervasive environments.

[3]The question about whether it is appropriate to do either is out of the scope of this discussion here.

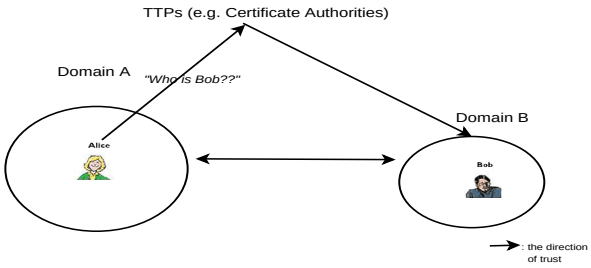

Figure 3.4: *Triangle trust diagram*

3.4 Localising the Trust Security Principle

Depending upon threats, different domains may have different assumptions. Consequently, they will have different local security policies, and correspondingly implement different security countermeasures. A foreign domain's policy is not always clear or available to local users. When the communication crosses domain boundaries, it is not only hard but (we shall argue) also unnecessary for a user from one domain to understand the precise security policy and mechanisms from other domains. As explained in section 2.3.2, trust is subjective, context-dependent and non-monotonic. Hence, when a lot of conventional systems use the term *trusted*, most of them really mean *trustworthy*. A TTP, e.g. the CA in a PKI system, is a typical example. However, trust is not trustworthiness [25], trust is *subjective*[4] but trustworthiness is a matter of objective fact. In other words, doing things right is not the same as doing the right thing. For example, a TTP may do things *right* ("trustworthiness"); however, even when it is obvious that this is so (and unfortunately it usually is *not*), it is still hard to convince every player that this is always a right thing to do in the first place. This is because different players may have different assumptions, probably because of their local domains' security polities.

The problem of this confusion is,

- "How do I know that you are doing your job properly". Users have no

[4]By *subjective*, I mean subject dependent or (in this dissertation's context) domain dependent.

idea if TTPs do their jobs responsibly. Even TTPs make a decision responsibly, users still have doubts whether this decision is for the right purpose or not, by the standard of their own domains. It is difficult for users to verify the decision during the communication if external CAs have not done their job properly by the standards of the local domain. Inevitably, trust transitivity problem [16, 25], the uncontrolled imposition of trust (e.g. the unfair compulsion to fully trust arbitrary CAs), surfaces. To rely upon foreign standards is a considerably irrelevant concern in the pervasive context. The primary issue is that external CAs have made a decision responsibly but unfortunately for a different purpose. Users cannot know whether it is right or not for their purpose from their viewpoint.

- "I do not consider A as a threat, because I trust A". The notion of *trust* is used here to rule out some threats. However, different users will look at a system from different perspectives. A trust assumption that is applied in one place cannot be passing freely over to all places in the loop.

Thus, we need a different [5] security design principle to guide security policy establishing trust for pervasive environments. This principle should be independent of any long-term stable and fixed trust infrastructure. I intend to lay to rest the assumption of relying upon external TTPs to achieve the users' security goal when the communication crosses multiple domains.

In the real world, we are most likely willing to trust ourselves. A local third party is also involved, mainly for efficiency reasons, e.g. buying tickets from a local travel agency. As this third party is local, we can easily verify their trustworthiness on a certain transaction. Similarly, a *localising the trust* security policy is proposed here for security over multiple domains. In terms

[5] *Different* in the sense of changing the way of thinking on current security policies. The policies themselves may be perfectly adequate in some cases.

of *localising the trust* for users, I specifically mean that the trust decision for a particular security purpose should be coming from users' own domains. This trust decision can certainly be delegated to some others users (from the same domain or from different domains) in different forms at some point. However, it has to be rooted from users themselves. We do not like external TTPs to make any final *commit/abort* decision for us. If we have to trust, we would rather trust a local authority's decision making or someone to whom we have freely chosen to delegate.

In the multiple domain context, it is difficult to pre-know the pervasive interactions that are going to occur. Hence, purely key-oriented or role-oriented security approaches will not work very well. A domain-oriented scheme is suggested here. As a matter of principle, a cross-domain security protocol should not force users to explicitly know precise semantics of security mechanisms in remote/foreign domains. Conversely, it ought to encourage each domain to be responsible for its own domain's security, further clearing up their own mess if necessary (as described in the transcript discussion for [39]). The objective of *localising the trust* security policy is a logical consequence of this understanding, particularly applied in the context of pervasive environments.

In figure 3.5, a user A and a user B are in different domains. The data flow will go through the user A, the user B, domain B's server and domain A's server. The *domain* is not just limited to a company, an organisation, a university or a department, which administrates many attached end-users. A user can also be regarded as a domain, particularly in the pervasive context. Furthermore, he/she can set up several domains according to the context of applications. More explicitly, for e-business applications, we can think of *domain A server* as a main e-business server (e.g. "www.amazon.co.uk") and *User A* as some image servers that are dealing with current purchase requests; *domain B server* can be some applications software running at users' home desktop and *User B* is more likely an agent, e.g. some trusted applications

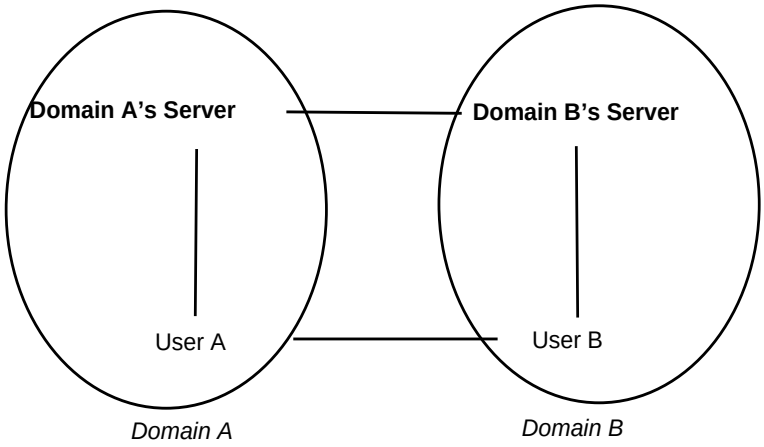

Figure 3.5: *Trust over multiple domains*

running at a PDA or laptop carried around. For the bank system, *domain A server* can be regarded as main bank servers and *User A* is existing bank client terminals (e.g. ATMs or credit card readers), *domain B server* and *User B*, still, can be home desktop and PDA respectively.

3.5 Security Countermeasures/Mechanisms

After identifying the threat and policy, a *security tool-kit* is deployed in the LoT framework. As explained above, I investigate the security issue in the multiple domain context from two ends, authentication and authorisation. As a consequence, LoT provides some example security mechanisms.

Authentication: authentication is also referred to as the *bootstrapping of trust*. It is traditionally a fundamental building block for security, supporting other security properties (e.g. confidentiality, integrity and availability) [119]. For most pervasive applications, there is usually no prior knowledge between pervasive users because they are most likely from different domains. In addition, another challenging difficulty to bootstrap trust here is the *unknown environment*. Numbers of hostile devices and other people may move around in the same environment where some pervasive interactions take place.

Therefore we need to be able to authenticate each other spontaneously (but not based upon our IDs) without a fixed trust infrastructure. Two channel authentication protocol is a quite straightforward example to achieve this, as it mimics the positive human context in the real world. The protocol takes advantage of the fact that it is easy to establish a low bandwidth but high data origin authenticity channel when participants are all in the vicinity [6]. Also, it explicitly demonstrates that mobility can somehow be useful to leverage the security [22] if we know what we are doing.

Authorisation: Authorisation is a significant method to *establish trust*, or *achieving access control*. Most resources or services in pervasive environments are provided by many different organisations and human users, even in one physical environment. Hence, it is infeasible to know which kind of access rights are required to access those resources in advance. In addition, pervasive environments require a flexible access control mechanism that does not rely upon the same centralised infrastructure fixed for each domain. Instead of the conventional identity-oriented [65], key-oriented [10] or role-based [106] approaches, a domain-oriented encryption-based access control (EBAC) is introduced by the LoT framework (see chapter 5). This primarily targets two typical scenarios when the access is over multiple domains.

1. Firstly, I will consider a scenario that two domains have established a trust relationship in some ways, or we can describe them as "friendly" domains. That is, domain B may be already *known* to domain A. For instance, the partners company A/company B are working on a joint-project. But nevertheless, a new assigned team member, Bob from Company B, would be a *stranger* to any team members from company A.

2. The second scenario is based upon the assumption that two domains

[6]Some applications may require further knowledge to establish the policy to determine whether it is appropriate to share a key with a device nearby. This is actually achieved by the authorisation end in LoT.

have not established any trust relationship or shared crypto-knowledge before (see section 6.5). However, a player needs to access some resources on a purely temporary basis, such as, a guest who would like to access some resources in the host's house. The guest has to be introduced to those resources by the host in order to access them. Thus, the user is "friendly" in spite of the fact that he/she is from a stranger domain.

Consider a simple multiple domain case, a player Bob from domain B wishes to access some resources in a remote domain A. Hence, Bob tries to get access permission from Alice (we can consider Alice as domain A's server). For most conventional approaches, for instance, Kerberos, Bob has to get a ticket from his own domain server before the interaction. Then, he submits this ticket to Alice. If Alice recognises Bob's ticket, she checks the access rights associated with the ticket to determine if Bob has the correct rights to access (compared with his access request). If so, Alice grants Bob's access request. Otherwise, Bob's access will be refused.

For LoT's EBAC, briefly speaking, Bob also needs to submit a form of credential to Alice. However, this credential is not used for Alice to make a final access decision. Instead, it contains sufficient (but just enough) information to tell Alice which domain Bob is from. This is achieved by a Profile Certificate. Bob submits his profile certificate to Alice during the interaction. Alice checks whether Bob's domain already has permission to access those resources. If so, Alice gives Bob an encrypted token which can be converted to a useful access capability only by the correct domain. Dual capabilities are being used to complete this conversion. Most importantly, Bob can only decrypt this token by authenticating himself to his own domain in accordance with his own domain's policy. If domain B is willing to delegate Bob that access, domain B's server will help Bob to decrypt the token. If domain B does not want Bob to access those resources in domain A, Bob cannot decrypt the token. Thus, Bob's access is granted only if the token issued by Alice is converted by

Bob's domain to a useful access information. As well as being authenticated by his own domain, Bob must also prove to Alice that he is in control of the hardware to which Alice has given the token. This process is described as domain-oriented Encryption-based access control.

Note that the discussion above is only conceptual. Specific mechanisms and protocols to achieve the required anti-properties (i.e. to guard against the threat) will be described in the remaining chapters. I am not claiming that those security countermeasures and mechanisms are ideal. Although novel, they are intended primarily as an existence proof. Other mechanisms may be better depending upon the precise application and context.

3.6 Conclusions

The LoT framework is a security infrastructure targeting the major concern of talking to *correct strangers* in the multiple domain context. It is based upon the concept of *localising the trust*. In LoT's world, every *player* is affiliated with its local domains. More significantly, domain security services/servers are urged to participate in local players' communications. LoT is designed to question the security issues in the multiple domain context from two ends, authentication and authorisation, and re-examine them in the light of the paradigm shift for security in multiple domains, i.e. pervasive environments.

Thus, I will examine the authentication end first in the next chapter. I will show the reason why a conventional meaning of authentication, i.e. ID-based authentication, is not what we want in pervasive environment, and propose a new two-channel protocol for *authenticating strangers* in LoT.

Chapter 4

Spontaneous Authentication for Pervasive Environments

Authentication is traditionally the fundamental building block to support other security properties. It is conventionally required to involve some form of proof of identity directly or indirectly. However, such strong (ID-based) authentication targets the wrong security requirements for pervasive environments, when humans are admitted into the loop.

Thus in this chapter, I focus upon the *classical* authentication problem, and argue that the desirable contextual information for different pervasive applications can be exploited to replace unnecessary strong (ID-based) authentication. First, I outline the basic security requirements for talking to correct strangers in the authentication context. Secondly, I consider how to positively re-frame the significant human context as a desirable security service for pervasive environments. Then, a two-channel authentication scheme involving using two-level protocols and human interactions is proposed. Finally, I describe two approaches from the DH-S3P protocol to address the public meeting threat model.

4.1 Basic Security Requirements in Pervasive Environments

The paradigm of pervasive computing [128, 129] introduces a new vision of an environment where users can communicate with resources regardless of the limitations of time and space. Human users are in the centre of this environment and are surrounded by numerous computing-capable devices. Those devices are embedded in daily objects of the environment, such as, Personal Digital Assistants (PDAs), cars, refrigerators, clothes, pens, etc., seamlessly pervading all parts of everyday life. They interact with each other spontaneously and (highly) dynamically, following human intension but without human awareness.

4.1.1 Talking to Correct Strangers

Authentication is originally introduced to guarantee that the communicating users are who they claim to be [19, 82, 93], implicitly or explicitly. It involves some form of identity directly or indirectly in most cases, as illustrated in 2.1.1. In conventional environments, the semantics of the human interactions that it supports have been largely ignored. Conventional ID-based authentication is always required to be strong enough to distinguish legitimate users from unauthorised users as a result. This is achieved mostly by relying upon challenge-response identification or interactive proofs involving TTPs. However, strong (ID-based) authentication is an inefficient, heavyweight task. It largely requires standardised infrastructures. As a result, to achieve strong (ID-based) authentication is hard.

We have seen a number of costly scenarios and systems being designed to authenticate communicating participants' identities, either universal names or logical identities. These aim to secure the association between legitimate users' identities and their resources. This kind of association attempts to

secure the entire communication from very beginning but is always subject to threats, such as ID-theft and spoofing. Generally, Bob masquerades as Alice not because Bob really fancies being Alice. Bob is rather interested in the resources or access rights associated with Alice's identity. Thus, my belief is that involvement of heavyweight identity-based authentication into protocols does not make them as secure as initially expected. Instead, such an approach opens attacks unnecessarily on identities in addition to incurring expensive costs.

An obvious cause of the difficulty to secure a pervasive environment is the spontaneous interactions between devices that meet for the first time in an ad-hoc manner. Very often, those devices are strangers to each other. In other words, they have not established any previous security association or crypto knowledge. They may not share a fresh secret key, or the corresponding public keys are unknown to each other, for instance. Arguably, traditional public key certificates could be implemented here. But this demands an accessible path to TTPs whensoever the protocol needs, not just to obtain the corresponding participants' certificates, but also to cope with more serious certificate revocation circumstances. Moreover, the communicating participants are most likely coming from different domains. Even if the recognition of certificates crossing domains is solved by chained negotiation among TTPs, there is no guarantee that the success of security policy checking in one domain will be propagated to other domains. It is increasingly frustrating for them to understand each other's security policy and further adopting the appropriate security mechanism. The unfair Trust Transitivity problem [16, 25] also indicates that the traditional pre-issued certificate-like authentication approaches have their own weaknesses for dynamic pervasive environments.

Thus, the security requirement in pervasive environments differs from the one addressed by conventional strong (ID-based) authentication schemes. Those

schemes deliver subtly the wrong security requirement for pervasive environments. The primary objective for pervasive users is not to find out the identity of another communicating participant to whom they are talking, "are they really who they claim to be". Instead, they intend to learn or validate whether the communicating participants have certain desirable contextual information for the particular interaction. Consider an example that Alice is going to print a document residing in her PDA to a printer nearby. She does not really care what the *nearby* printer's name is. The essential concern in terms of the authentication is if the document is sent from Alice's PDA to the printer she is looking at [1]. Here, *nearby* is the contextual information for the printing job required by Alice, "I do not care who you are, but I do care you are the one just next to me" [2]. Alice does not want her private document sent to any other printers in the same room, or worse, further away, without her notice, as the result of intentional attacks or unintentional misconfiguration.

I am motivated by the context of talking to correct strangers in pervasive environments from the beginning of this research. A typical example application is given below in this chapter. It is important to pinpoint that the name or ID of the *user* (e.g. a person, a device or a process) is meaningless as far as this form of authentication is concerned.

4.1.2 Motivating Threat Model - Public Meeting 1

Company A and company B are doing a joint business project together. Alice and Bob are the marketing managers for company A and B respectively. They meet each other for the first time in a public conference room, which contains many other people with many computer devices. Alice wishes to send

[1] It is assumed that Alice trusts this nearby printer will not pass the document to other printers or devices, perhaps because she sees a particular manufacturer's tamper-evident seal on the printer.

[2] Again, we should also take the "local" security policy into consideration, particularly if Alice and the printer are in the same domain. For example, the local security policy may restrict Alice to print sensitive document only on authorised printers.

a private project plan document m to Bob in some wireless way, for instance, the mobile ad-hoc communication established between their personal devices DRD_{alice} and DRD_{bob}. In this dissertation, DRD_x, as Digital Representative Devices, refers to a personal computer device with wireless communications capability and reasonable computational resources for a human user x. Notice that, for the purpose of spontaneous authentication, I assume that *Bob* is the person who Alice has already known to be the *correct stranger* in this example. Thus, the *correct stranger* here explicitly means their personal devices have not established any secure association before. A more complex scenario will be discussed in the next chapter.

A major security concern in this case is numbers of hostile devices and other people moving around in the same conference room. Alice does not want the private document sent to another device called DRD_{bob} (intentionally or unintentionally) held by someone else in this room, instead of by Bob who is standing next to her. Thus, the main threat here is whether the document is sent from the DRD_{alice} held by Alice to the DRD_{bob} held by Bob who is standing next to Alice.

Notice that DRD_{alice} and DRD_{bob} have *not* set up a security association before this public meeting. It is well understood that the wireless communication channel is vulnerable to both passive attacks and active attacks. An attacker within a wireless radio range can easily eavesdrop or modify the document. Furthermore, the attacker can even masquerade as Alice (precisely DRD_{alice}) or do a man in the middle attack. Another threat in the public meeting model is that the document transmitted between the DRD_{alice} and DRD_{bob} can be overheard and possibly modified.

Alice and Bob might intuitively choose an arbitrarily reasonably long value[3] as a shared key phrase to encrypt the transmitted document. This only works if the surrounding area is guaranteed to be secure, e.g. in a locked private

[3]It should be long enough to be invulnerable to exhaustive search.

meeting room, in a Faraday cage. Otherwise, for circumstances like the public conference room, an attacker Moriarty can easily obtain the key by either peeking over shoulder or a hidden CCTV, enabling him to violate the secure communication between Alice and Bob's devices. This is similar to what we have experienced in the current Chip & PIN credit card approach deployed in the UK.

Consequently, the ultimate security goal is to establish a spontaneous secure communication between DRDs being held by Alice and Bob, who are standing next to each other, respectively. I will give my approaches in 4.3.

4.2 Significant Human Context in Pervasive Environments

Nowadays, pervasive environments are characterised by the achievement of computer-invisibility, people communicating by means of the presence of physical visible devices but without noticing their existence. It is very clear that the human context is the distinctive property for pervasive environments. Explicitly, it is desirable to transform security techniques into the new human-based philosophy for pervasive environments.

4.2.1 Positive Human Context

Among the more serious threats which make cryptosystems fail in the real world are human implementation errors and management failures [3]. Superficially, limiting human influences on computing systems is usually a basic discipline to guide security protocols design, especially for authentication protocols in conventional environments. This is considered reasonable because of human unpredictability, including dishonest or incompetent behaviours. Hence, we always worry that the involvement of human context would mess up the security dramatically as many cases witnessed in Mitnick's book [87]. Humans

are invisible in conventional communication contexts such as Internet-based computing. Computer devices follow human instructions, but ignore whether these instructions are appropriate to the tasks or come from the right human. Thus, strong (ID-based) authentication is always required to ensure that a tracing step can be followed if something is going wrong.

Michael Roe in his Ph.D thesis [103] shows that what is regarded as a security threat in one context may become a mechanism providing a desired security service in a different context. We have to look at the context to decide whether something is a threat or a security service. Human influences are usually negatively considered as *threats*. But nevertheless, the communications in pervasive environments always occur in a highly dynamic and spontaneous way. This results in the infeasibility to have proper pre-computational resources configured for a particular interaction. For those interactions between human and human, human and devices, only the human has the contextual knowledge about the forthcoming interactions. For instance, we often have pre-decision ("This is the one to whom I am willing to talk"), and physico-spatial knowledge ("Yes, I can see this is the one I am going to talk to") for pervasive applications. Thus, the human context cannot be simply considered as a threat because it is the distinctive property for pervasive environments. I shall attempt to positively re-frame knowledgeable human influence as a desirable security mechanism in the pervasive context. On the one hand, human users will not worry about the details of pervasive interactions. On the other hand, they ought to be encouraged to interact with the devices and environments more positively, leveraging the ultimate security goal.

Thus, in a similar manner to Roe's *threat/service duality*, I argue that the positive human context is the distinctive security service for pervasive environments. Conversely, failing to recognise the positive human context is a threat in the pervasive context.

4.2.2 Minimise The Reliance Upon Trustworthiness

A *maltrust problem* is defined if humans abuse *trust* gained from other humans. Most conventional schemes are built upon computed credentials [4] from computing devices, intending to solve the *maltrust problem*. A typical example in the real world is the current Chip and PIN credit card approach. It intends to shift the final jurisdiction from human verification (signature recognition) to computed authentication (system verifying PIN matching). These schemes, however, have not achieved better security performance because the essential *maltrust problem* has not been solved as it was expected to be. Instead, it is simply reproduced from the human-human domain to the human-device domain. Consequently, increasing human reliance upon computer devices with the seamless interactions between humans and devices in pervasive environments is in fact compounding the problems caused by *maltrust*.

The principle of my proposal is based on a *Need-to-Know* policy [5]. This policy is not new, and was originally produced in a military context and classically applied in access control systems via minimising access rights. Note that in this approach it is critical to relate authentication explicitly to access control, because the primary purpose of authentication in a *Need-to-Know* context is precisely to determine (minimal) access rights. Here, we transfer this idea to authentication protocols and introduce a *minimise the reliance upon trustworthiness* principle to balance trust coming from human and computer device domains.

For *DRDs*, it is a high-cost and complicated job to deal with unpredictable confusions by depending only on computational results. Likewise, each *DRD* cannot simply be assumed honest, competent, and willing to perform expensive

[4] Computed credentials are bit-pattern which are solely calculated by computing algorithm behind the scene.

[5] Regardless of how freely we wish to make resources available, it is dangerous (from the integrity and audit dimensions of security) for users to hold capabilities which they do not even intend to use, as explained in the *principle of least privilege* [40, 105]

tasks strictly. So it is unfair to establish trustworthiness from authorities' assurances (due to an obvious trust transitivity problem) and it is worse to rely entirely on the results of computations performed by computer devices with no human interaction (another expression of trust transitivity). For instance, when customers withdraw money from an ATM, they cannot ensure (or even verify) that the ATM will implement security policy checking correctly (but interestingly, both banks and customers usually assume ATMs will do so).

As I pointed out above, positive human involvement is necessary to the security of pervasive computing. Introducing human context into security protocols has the potential to guide pervasive computer devices to deal with complex security requirements effectively. I have always been inspired by a comment of Mark Weiser, the father of ubiquitous computing, in his well-known paper [128]:

> *"There is more information available at our fingertips during a walk in the woods than in any computer system, yet people find a walk among trees relaxing and computers frustrating. Machines that fit the human environment, instead of forcing humans to enter theirs, will make using a computer as refreshing as taking a walk in the woods."*

4.3 Two Channel Authentication Protocols

4.3.1 Leave Strong Authentication Behind

Strong (ID-based) authentication does not do us any favours when we attempt to secure pervasive environments. It is hard, introduces unnecessary attacks, and more significantly, it is not what we want to solve the problem, as explained in 4.1.1. Hence, I intend to ask a logical question, "why not just break the association between identities and authorised resources or access rights?" In other words, "Strong authentication is not always necessary in all

circumstances". Breaking this link can enable us to achieve:

- Privacy and Identity Protection: one desirable consequence is that a raw identity will not be valuable any more so that ID-theft or spoofing makes no sense at all. Hence, it protects identity indirectly. Moreover, excluding identity information in the protocols will satisfy human privacy requirements.

- Data Uncorrelation: another exciting gain is to erase correlation among all input/output data streams with respect to entities. Such uncorrelation makes many active attacks more difficult.

Some may disagree, and argue that unacceptable risks arise by dropping strong (ID-based) authentication from protocols, particularly in the sense of talking to strangers in pervasive computing [6]. It is indeed risky to talk to strangers; however, such risks do not arise from whether protocols are equipped with strong authentication or not. Instead, these risks are coming from the requirements of the applications themselves, i.e. talking to strangers. A similar philosophy applies in human daily life. If Alice trusts Bob whom she has not met or trusted before, then Alice has to risk the possible consequences. Protocols without strong authentication will not necessarily weaken security performance compared with the ones which have. Conversely, they can eliminate the threats accompanied with unnecessary strong authentication.

4.3.2 Spontaneous Authentication with Two Channels Protocol

Despite the feasibility of leaving conventional strong authentication behind in pervasive environments, distinguishing legitimate users from unauthorised

[6] Another likely concern is how to provide an *audit trail* without requiring communicating parties' identities as part of authentication protocols. Although a discussion on audit trails is out of the scope of this dissertation, I also briefly highlight a possible way to support an audit trail in the LoT framework (see the footnote 3 on 107).

users is still an issue. Here, I propose my mechanism *Spontaneous Authentication* or human thinkable authentication [7]. Thinking is a distinctive ability in human behaviours, which is unlikely to be exhibited by any computational device[8]. The spontaneous "thinkable" authentication protocols with the human self-determination contrast with the traditional "computable" authentication protocols which involve no distinctively human agency.

In order to achieve this goal, it is expected to impose necessary *tolerances* to executed protocols. There is no entirely *transparent* trust in most cases for pervasive applications. *Transparent* trust in this dissertation means that two entities have established a trust relationship before (e.g. share a secret key) or have been introduced by knowledgeable authorities (e.g. holding corresponding certificates). The *tolerance* property should be understood differently depending upon applications. For instance in pervasive environments, the basic wireless RF channel does not have high data origin authenticity. Instead of expending too much cost on making an RF channel with that characteristic, it is desirable to make the protocols tolerant of this limitation. More precisely, another out of band channel is assumed with the required characteristic (high data origin authenticity in this case). Semantically, such an out of band channel is quite similar to a location-limited [13], or empirical [33] channel. It is a relatively low bandwidth channel compared with high bandwidth RF channel. It is subject to passive attack but not to active attack. Therefore, the ad-hoc communication participants can be assured that the data on this channel does really come from their counterparts. It could be realised in many ways between human-human and human-device, for instance, physical contact between the devices, a close range infra-red link, or one device displaying a number on the screen which is typed into another device by the human user,

[7] Consider the human thinking ability, this is to put human in the authentication loop.

[8] The possibility of devices which can pass the *Turing test* [125] is beyond the scope of this dissertation, but arguably such devices should be regarded as human users rather than as DRDs from the cyber rights perspective.

and so on. Essentially, the human context (e.g. hearing, monitoring) is required at this level. We call these human contexts *human self-determination*. Knowledge is said to be *human self-determination* if the knowledge is acquired via necessary human contexts, such as, hearing a tune or seeing a display.

Thus, two levels of security mechanism are introduced to incorporate human self-determination knowledge into authentication protocols due to the existence of two channels. It allows tradeoff between trustworthiness in both sectors (human-human and human-device).

1. A *Plausible (but unreliable) trust* (or PT) protocol is used in the high bandwidth channel with the necessary security *tolerances*. The RF channel is subject to both passive attacks and active attacks. Thus, *tolerances* here explicitly means that the main purpose of this protocol is to stop passive attacks. Trust gained from the PT protocol run is plausible, but not reliable for active attacks which occur in the high bandwidth RF channel. I will give an example to explain what I mean by this form of trust in 4.4.1 below.

2. A *Reliable trust (acquired through human self-determination)* (or RT) protocol is called to achieve higher levels of assurance (e.g. high data origin authenticity) with the assistance of the out-of-band channel. This comes with the mandatory interaction of human context, e.g. monitoring, hearing, recording, etc., depending upon the choice of out-of-band channel. This is expected to gain an equivalent outcome to that which strong authentication schemes achieve in conventional environments. The protocol's run is completed as success of a *human trust-based decision process* [77].

These two protocols work together to support the *Spontaneous Authentication* with two channel protocol argument. Some existing protocols (some of them are listed in section 2) can be substantially adapted to the *Spontaneous*

Authentication hypothesis, e.g. physical contact authentication in Stajano's *Resurrecting Duckling* [120, 121, 13], entity recognition module [112], authentication starting from weak secret agreement protocols and applications, and other contextual attributes (i.e. time, temperature, services, locations, specific transactions) stated in [32].

After implementing the PT and RT protocol, eventually, the human users must further be assumed that a fresh session key has been shared between the correct ad-hoc devices. More significantly, no other device or person can know this session key.

4.4 Example Protocols

The Diffie-Hellman (DH) key exchange protocol [41] is the classic solution to the key agreement problem in a decentralised environment. However, the traditional DH key exchange protocol relies upon the assurance of integrity in the high bandwidth message exchange channel. Such an assumption can barely be achieved in the pervasive context when the wireless RF channel is deployed. Correspondingly, a prior context (e.g. a password or a nonce) has been involved in many protocols [8, 9, 28, 50], mainly targeting the man-in-the-middle attack. It is still a problem for most pervasive applications, for instance, the public meeting threat model (in the 4.1.2) which I am investigating in this chapter. The prior context exchanged between Alice and Bob, or their hand-held devices, is vulnerable in an open (even hostile) environment (e.g. the public conference room). Anyone who successfully obtains the prior context can break the entire authentication protocol.

I will give two approaches built upon a basic DH-S3P protocol [28]. They require both significant human context and two channel authentication to address the problem. Consider the public meeting scenario, assume that,

DRD_{alice}, DRD_{bob}: two private DRDs held by Alice and Bob respectively,

which have monitor screens (e.g. PDA) and sufficient computational resources;

a, b: random numbers generated by DRD_{alice}, DRD_{bob} respectively. They have to be strong enough to be the discrete logarithm (DL) exponent.

Also, we assume generator g, large prime modulus $q = 2p + 1$ for prime p, and one-way hash function h are publicly known. Random numbers a and b generated by DRDs must be strong (i.e. long enough to be invulnerable to exhaustive search) as well as hard to predict. A full discussion of assumptions as preconditions not specific to the mobile ad-hoc context is given in [28].

4.4.1 Basic Approach

The first basic approach is based upon our early work [78]. We assume that Alice and Bob agree a shared weak secret k in the meeting. It may, for instance, be a password which might be reasonably short. Alice could choose a value of k and secretly tell Bob before the following processes take place.

- Setup Phase in PT protocol:

  1. Alice initiates request by inputting k into DRD_{alice}, generating a random number a and demanding DRD_{alice} to set up mobile ad-hoc communication with DRD_{bob},
     $DRD_{alice} \rightarrow DRD_{bob}$: X_A
     where, $X_A = g^a + k \bmod q$.
  2. Bob inputs the same k which he obtains from Alice, into DRD_{bob}. DRD_{bob} generates random number b and responds,
     $DRD_{bob} \rightarrow DRD_{alice}$: Y_B
     where, $Y_B = g^b + k \bmod q$.

- Marking Phase in RT protocol:
  After the setup phase, neither Alice nor Bob has sufficient knowledge to determine whether messages are coming from a genuine device or from

malicious ones.

However, DRD_{alice} and DRD_{bob} both generate,

$$g^{2ab} \bmod q = (s|n),$$

where, s is a session key and n is a nonce (with reasonable length)

Then, both devices calculate,

$$n_1 = h\{n\}$$

and show n_1 on the screens in some graphic form[9]. Here, I use a slightly different message exchange sequence from the one described in [28].

Now, note that in addition to the RF channel, a human visual checking channel is deployed as an out of band channel in this phase. Thus, Alice and Bob have to check the hash images displayed on both DRD_{alice} and DRD_{bob}'s screens. Mutual authentication is eventually completed only by Alice and Bob observing matching hash images. By then, Alice agrees to send m encrypted under the session s which is only known by DRD_{bob}. Both of them will abort the interaction otherwise, if the hash images are not matching.

Developing positive human context into an authentication protocol allows weak confidentiality to be boosted into strong confidentiality, which is provided by subsequent session key use. In contrast with the original DH-S3P protocol, the security of this approach only partially depends on the continuing secrecy of the shared password k.

Admittedly, it is very difficult to guarantee that no human nearby Alice and Bob will peek at the process in order to get password k. An attacker Moriarty may obtain the weak secret k. He would, however, need to get

[9]Perhaps similar to those used in CAPTCHA [126, 127]. Such graphics are sufficiently easy for humans to distinguish, but hard for computer devices to spoof, to replace hash function bit-values.

k early enough (before the message 1) in order to perform the man-in-the-middle attack. Note that the final hash value n_1 used for human verification is essentially truncated from the $h\{g^{2ab}\}$. Hence, if the bit length of n_1 is too short (say 16 bits only), Moriarty (his personal device DRD_M) can actively intercept the wireless communication (as shown in [50]),

1. Moriarty replaces the message 1 with $g^{a'} + k \bmod q$, for some random value a' chosen by himself.
   $DRD_{alice} \rightarrow DRD_M$: $g^a + k \bmod q$,
   $DRD_M \rightarrow DRD_{bob}$: $g^{a'} + k \bmod q$,

2. Bob generates his DH value (g^b) and sends,
   $DRD_{bob} \rightarrow DRD_M$: $g^b + k \bmod q$,
   At this point, Bob computes $g^{2a'b} \bmod q = (s|n)$ and hashes the last 16 bits ($n'_1 = h\{n\}$) as the image;

3. Moriary computes the $X' = g^{2a'b} \bmod q$ as well. Then, he chooses a set of random values b'. For each b', he computes the $Y' = g^{2ab'} \bmod q$ and compares the last 16-bit of $h\{X'\}$ and $h\{Y'\}$ until the matching is found. Moriarty sends this particular b' to Alice,
   $DRD_M \rightarrow DRD_{alice}$: $g^{b'} + k \bmod q$,

4. Alice computes $g^{2ab'} \bmod q = (s'|n)$ and hashes the last 16 bits ($n'_1 = h\{n\}$).

5. Finally, Alice and Bob will find the matching images displayed in the screens although they do not share a session key. Moriarty will get the m when Alice encrypts the message with the key s (the head of X'), which is in fact shared with Moriarty. Bob encrypts any message with s' (the head of Y') which is also shared with Moriarty. It is feasible to do so in a short time if the attacker has significant (pre-)computational resources.

Thus, the bit value of n needs to be long enough (at least 48 bits according to [50]).

However, this approach requires the attacker to engage in the protocol early enough. If the attacker only gets the k after the messages 1 and 2 transmission between A and B. The attacker cannot get the value of g^{2ab}, because he cannot solve the DL problem. Thus, devices which behave as "man-in-the-middle" still cannot know the private message m. Even if, Moriarty obtains one of the DL exponents (namely, a or b) as well somehow, which means he might be able to re-produce the hash image. However, the k is used to calculate the decryption key for the message decipher. In other words, the k is used to calculate subsequent protocol values. Moriarty has to commit this exponent value before he learns the k. Hence, it would be too late for Moriarty to interfere with the communication between Alice and Bob. One possibility to attack this approach is to repeat the message (1) and send it to DRD_{bob}, particularly when Alice and Bob would like to exchange another company plan m_2 (after a short period) but from DRD_{bob} to DRD_{alice}. To block such a replay attack, necessary freshness can be provided using a nonce. More significantly, this change addresses the man-in-the-middle attack for the short truncated hash value described above. This will be examined in the next approach.

4.4.2 Generic Approach with two Channel Protocol

I will show a generic approach using nonces. We assume that, n_a and n_b are the nonces generated by DRD_{alice} and DRD_{bob} respectively. It is assumed that both nonces are generated by the crypto-modules inside the devices. The attacker cannot see what it is going on inside the crypto-modules even if spyware is running inside the DRDs. Thus, those nonces are invisible to the attacker when the interaction occurs.

Two physically different channels are used in this approach. The notation, $X \rightarrow_C Y : Z$ represents the data Z is transmitted from X to Y in the

channel C.

In the public meeting model, two channels, one of which is RF channel and the other is an out-of-band channel (OoB), an infrared link for instance, are deployed.

1. DRD_{alice} generates a random number a and a weak nonce n_a, setting up mobile ad-hoc communication with DRD_{bob},
   $DRD_{alice} \rightarrow_{RF} DRD_{bob}$: X_A,
   where, $X_A = g^a + n_a \bmod q$,

2. DRD_{bob} also generates a random number b and a weak nonce n_b, responding with,
   $DRD_{bob} \rightarrow_{RF} DRD_{alice}$: Y_B,
   where, $Y_B = g^b + n_b \bmod q$;

3. At this point, the second, out of band, channel (denoted OoB) is used,
   $DRD_{alice} \rightarrow_{OoB} DRD_{bob}$: n_a,
   $DRD_{bob} \rightarrow_{OoB} DRD_{alice}$: n_b,

4. The final stage is similar to the one in the basic approach, using necessary human self-determination. DRD_{alice} and DRD_{bob} both generate,
   $g^{2ab} \bmod q = (s|n)$,
   where s is a session key and n is a nonce. Both devices show the hash images in the screens. Alice and Bob have to check if the hash images are matching.

   For the final stage, alternatively, we can assume that DRD_{alice} and DRD_{bob} both compute, $g^{2ab} \bmod q = (s|n_1|n_2)$,
   where, s is still a session key. The nonce n_1 and n_2 generated from the calculation ($g^{2ab} \bmod q$) have the same constant bit-length (let us say 50 bits). Hence, instead of checking hash images, Alice and Bob are doing the following step,

- $DRD_{alice} \rightarrow_{RF} DRD_{bob}$: n_1,
- $DRD_{bob} \rightarrow_{RF} DRD_{alice}$: n_2,

Now, the crypto-modules inside the devices DRD_{alice} and DRD_{bob} check whether the received value (n_1 or n_2 respectively) matches the one obtained from ($g^{2ab} \bmod q$). If both crypto-modules can find the match, Alice will send the message encrypted under s. Otherwise, they announce an error [10].

The attacker Moriarty cannot influence n_a and n_b during the communication. As explained previously, the OoB channel is subject to passive attacks but not for active attacks. It is possible for the attacker Moriarty to learn n_a and n_b but only after step 3. In other words, it is too late for Moriarty to attack truncated values, even he has sufficient pre-computation resources. Hence, the man-in-the-middle attack is infeasible. Moreover, n_a and n_b are revealed via the out-of-band channel after the stranger devices are committed to the messages transmitted over the RF channel. This approach is similar to some authentication protocols (for mobile ad-hoc or pervasive computing) introduced in chapter 2. However there is a major difference in terms of the security semantics. In this two channel approach, the messages transmitted via the second channel are required to calculate subsequent security values, rather than simply to do equality verification. Hence, a lazy or unauthenticated player will not cause a security breach.

Both approaches enhance the conventional DH key exchange protocol, particularly for the purpose of talking to the correct strangers in pervasive environments. Essentially in both approaches, the conventional DH protocol consists of the first two messages (with k or n_a, n_b set to zero). The significant human context is introduced to complete the authentication process for pervasive environments.

[10] This alternative step was suggested during a discussion with Prof. Bruce Christianson. For detail refer to the paper [26].

4.5 Conclusions

The protocols in this chapter highlights the usefulness of exploiting contextual information in pervasive environments instead of replying upon conventional strong (ID-based) authentication scenarios. The concept of utilising out of band channels can promote significant human context to pervasive environments. This approach is spontaneous and independent of the infrastructure. Moreover, it implies an efficient solution to the puzzle of talking to the correct strangers for most pervasive applications. But what authentication has to achieve depends upon what can go wrong.

For the threat model targeted in this chapter, human users Alice and Bob are assumed to be known to each other as the correct strangers already. The main concern is the spontaneous authentication between devices held by the human users. In the next chapter, I will relax this assumption and investigate the security requirements under the absence of human trust (e.g. Alice and Bob do not know each other). Correspondingly, I will explicitly relate this problem to the classical access control scenario because the major purpose of authentication is to determine (minimal) access rights.

Chapter 5

The Domain-Oriented Approach for Access Control Over Multiple Domains

The most important aspect of the LoT framework is the *authorisation* end. The concept of achieving access control in the multiple domain context is *syntactically* similar to the one witnessed in conventional environments. *Semantically*, however, access control over multiple domains is different because it requires an effective approach that does not rely upon the same fixed trust infrastructure for each domain. The conventional approach to this problem is to introduce a globally trusted TTPs. However, I wish to encourage local domains to take part in their users' interactions as a result of the *localising the trust* security policy. Thus, I propose a domain-oriented encryption-based access control scheme in this chapter. This scheme is the basic foundation underpinning the LoT access control mechanism.

This chapter is focused on an architectural description and some associated issues (i.e. delegation, revocation) for encryption-based access control (EBAC) in the multiple domain context. It provides a conceptual basis for the following chapter. This chapter begins with the highlight of the advantage of developing a domain-oriented viewpoint for access control in the multiple domain context. Section 5.2 describes a new form of certificate, Profile Certificate, which is used

only for conveying information between users and their associated domains, but not for making a final commit/abort decision for users. When a user from one domain attempts to access some resources in other domains, *delegation* and localised authentication will occur to achieve authorisation. This will be analysed in Section 5.3. Section 5.4 presents the architecture of EBAC constructed in a novel way. This chapter ends with a discussion on some key features of achieving revocation in the EBAC scheme.

5.1 Domain-Oriented Access Control Method

5.1.1 Access Control over Multiple Domains

The trust establishment problem is a derived form of the classic *Access Control* problem [62]. This can be seen from the simplest example, "can Server S allow User U performing operation E on resource R". From the resource server's (S) perspective, a typical solution to this problem involves two logical steps.

1. Step 1: *Authenticating* the user U. Conventionally, this step is achieved via the verification of the user's ID, cryption-key (i.e. public key), or a role. These approaches are (very often) referred to the *ID-oriented*, *Key-oriented* or *Role-oriented* methods for access control. Inevitably, the association between a user and a crypto-key, the crypto-key and access rights, a user and a role is guaranteed by the presence of TTPs, because the user U may be "unknown" to the resource server S. This idea can simply be illustrated as a triangle diagram in figure 5.1. The resource server has to reply upon TTPs to make a security decision, in order to achieve its own security goals.

   It is fair to say that the resource server totally depends upon a fixed trust infrastructure [44], such as CAs and directory servers, to make a correct decision here. Relying upon TTPs to make a final commit/abort security decision is sound and workable for one single domain because

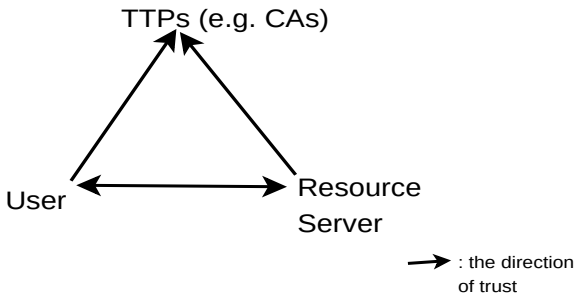

Figure 5.1: *Trust triangle*

the local security policy and resources are clear to all entities.

However, pervasive interactions almost always involve users from many possible different domains spontaneously. A foreign domain's policy is not always clear or available to all users. Therefore, it is difficult for users to verify the access decision during communications if external TTPs have not done their job consistently with the standards of the local domain.

2. Step 2: *Authorising* the access request for U according to the access control policy, if step 1 is completed.

   Access control policies can be controlled by either external authorities or resource servers themselves[1]. If external authorities manage access control policies, they need to know if any access control policy change occurred in the resource domain because they are unlikely to be resource owners. However, this mandatory type of access control faces additional challenges in any *genuinely* multiple domain context. It is problematic to obtain those observations without the support from a global fixed trust infrastructure in pervasive environments. Moreover, a pervasive user might not know which kind of access right is needed to access a certain resource. Most of all, imposed by the *localising of trust* security

[1]These scenarios are also known as Mandatory Access Control and Discretionary Access Control [95, 107] respectively.

design principle in this dissertation, access control policies ought to be administrated by the resource owner. Unlike mandatory access control, LoT does not have a central administrator to monitor the distribution of access rights. The proper access rights will instead be *delegated* to other users *freely*. I shall discuss *delegation* in detail in section 5.3

The syntax of the access control mechanism in LoT is similar to the one for most conventional access control systems. I will still look at access control in multiple domains from these two steps, *authentication* and *authorisation*. However, due to the different requirement, specifically for the *authentication* step, the semantics of achieving access control changes. Thus, for the authorisation end in LoT, I will pay more attention to step 1 in the access control process and argue that the multiple domain context requires a more effective way to achieve it.

5.1.2 Domain-Oriented Approach

Access requests across domain boundaries, i.e. pervasive environments, require a flexible access control mechanism that is independent of a long-term stable and fixed trust infrastructure [35, 45, 121]. As indicated in chapter 3, LoT's view of the world is *domain-oriented*. Explicitly, domains are responsible to make the security decisions for their local users with respect to local policies.

For achieving access control over multiple domains, I argue that the user should be essentially *authenticated* by his/her local domain, rather than being *authenticated* by a remote resource server's domain. To explain this, I develop my argument by analysis of the three statements below.

As illustrated in figure 5.2, assume a Ph.D student, Bob, from university B, intend to use the service S (e.g. fax machine, the Internet) in a public conference centre.

1. Statement 1:

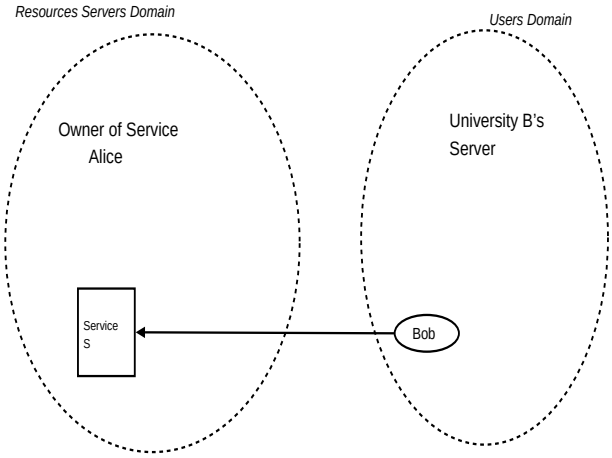

Figure 5.2: *Alice is the owner of the service S, and she will not let Bob access S unless Bob proves he is a current Ph.D student from university B. We can regard Alice as a resource domain and university B as a users domain for instance.*

Bob: "my name is *Bob*, a Ph.D student from *University B*." handing over the student card possibly;

Alice: "who is *Bob*? is this credential (i.e. the student card) from *Bob* really belonging to you?"

2. Statement 2:

   Bob: "This key K has the right proving I am a Ph.D student from *University B*." We also can substitute K with a capability (e.g. [55, 71]), or a delegation token (e.g. [49, 118]).

   Alice: "is K still valid? has it been revoked by university B already?"

3. Statement 3:

   Bob: "I am a Ph.D student from *University B*, and I would like to access the service S."

   Alice: "OK, this is a capability to access S, however, it is locked and can only be unlocked by *University B*."

   Bob calls university B to get an appropriate ticket to unlock the access capability by authenticating him to his own university as an valid Ph.D

student.

For both statement 1 and 2, the association between the user Bob and its ID or key is essentially defined by the authentication mechanism in *University B*. It is frustrating for Alice to understand the precise semantics of the authentication mechanisms used by University B. Thus, inevitably, external authorities have to get involved here.

Statement 3 is different [2] as Alice does not care who Bob is, or what Bob's key is. The only thing she wants to know is whether university B considers this access requester as a Ph.D student or not at this moment. For this requirement, the administrative server from *university B* (rather than Alice herself) is the most suitable place to make a decision. Hence, essentially, Bob is required to *authenticate* himself to his own domain, university B. The term *authenticate* is used here at a more generic level. This form of authentication can be done by using either a classic user authentication-focused mechanism or a more access control-focused tool.

To allow domains to authenticate their local users is more efficient than implementing authentication across domains. Alice does not need to know the precise semantics of security mechanisms, such as authentication, in University B. In essence, the domain-oriented approach for access control in LoT intends to reduce the difficult problem (access control over multiple domains) to two considerably easier problems (user authentication in a single local domain and remote authorisation between two domains) that we are more confident to deal with.

This domain-oriented approach for access control is well suited for the LoT's world because it ties a user's claim closely with his/her affiliated local domain. It follows the *localising the trust* policy by letting domains authenticate

[2]This approach may seem "unappealing" in practice. However, I intend to build up from this simple statement to show the semantics of the domain-oriented approach for access control over multiple domains.

their local users if authentication is necessary. More significantly, it encourages domains to get involved with their local users' interactions. Hence, we need to have a form of information showing which domain the user is from. The association between a user and its domain is quite straightforward for an email system. Interestingly, most Identity-based Encryption systems use the email system to explain their semantics. A user's email address clearly indicates the association between a domain and a user, for instance, *LJ@herts.ac.uk*, which contains the knowledge that "the user LJ is coming from domain *herts.ac.uk*". It is another issue to verify the authenticity of the domain, "is there really a domain named *herts.ac.uk* (and is it the university or not)" or the relationship, "is LJ really AT *herts.ac.uk*". Whether such an association between users and domains is true or not is actually controlled by the encryption-based access control mechanism which will be introduced in section 5.4. For now, let us focus on the mechanism for relating users to corresponding domains.

5.2 Profile Certificates

For PKI and other similar schemes involving TTPs, an advantage is that authenticating a domain is easy (compared to authenticating a large number of individual end users) [24]. Early Identity Certificates (i.e. X.509 [65] and PGP [132]) operated to guarantee the authenticity of a certain user's public key by associating it with a user's ID. However, global identity (i.e. globally unique user IDs) is a strong assumption and more significantly it does not really solve the correct problem (that we would like to be solved).Thus, SPKI/SDSI systems [99, 43] have defined a more flexible and security sensitive form of certificate, the Attribute Certificate. It breaks unnecessary binding between identities and public keys. It works with an access control policy by embedding key holders' privileges into certificates.

For the Identity-based Cryptography (or IBC), it is easy to authenticate

an individual user, as the users' IDs are the significant component for their public keys. Most IDC systems do not require any forms of certificates by assuming that each domain's validated public key is always available. I intend to relax this assumption a little bit more.

I will re-define traditional certificates and introduce a new form of certificate, Profile Certificate (PC), for implementing the Encryption-based Access Control mechanism. By way of contrast to existing certificates, the PC here does not act in a decision making role for users. I do not mind having CAs in a security system, but they shall be utilised correctly for the right purpose instead of effectively acting as users' decision makers. The sole purpose of having PC here is to attach a user to a domain for a particular purpose (note that users may have a presence in several different domains for different purposes.). Semantically, it is analogous to an email address for a user. It is not necessary to have CAs in each domain or hierarchical/chained infrastructure. Instead, a certain number of (optimistically) trusted authorities called Profile Certificate Authorities (or $ProCAs$), are available distributively, and no trust transitivity subsists among them.

A $ProCA$'s responsibility is not to supply any crypto knowledge. They intend to provide the information that a certificate holder is from a certain domain. Whether such information is correct or not is subsequently controlled by the domain itself rather than the $ProCA$. We know that the entire security of the user domain can be compromised if the traditional CAs make a wrong decision. On the other hand, if the $ProCA$ certify the wrong relationship between users and domains, it does not matter (except for performance) because the decision is *ultimately* managed by the local domain. It is an efficient approach as the $ProCA$ do not need to understand the semantics of the requests from users. The $ProCA$ only provides hints of where to get started.

Public keys of those $ProCAs$ are assumed to be always available. Again, the IDs or names of domains or users do not have any security-related meaning

in the system. The core elements for Profile Certificates (PC) are,

1. **Owner's public key**: instead of naming an entity, the owner is identified by the public key.

2. **Issuer's public key**: identifier of the issuing $ProCA$. The $ProCA$ embeds its own public key into the owner's profile certificate. Hence, all verifiers of a profile certificate are able to check if a correct $ProCA$'s public key is used.

3. **Profile**: the outlined security view for the owner, associating the owner with corresponding domain. The way of profiling domains/users will vary with respect to the context of applications, in addition to the domain's local policy. It can depend upon the role, the location, the date, and so on. Likewise, if a local domain's security policy is really satisfied with having a local unique identity/name as a profile, it is not that bad as long as the policy can be kept locally. However, I will not encourage to do so considering the significant expense on the practical implementation. Taking revocation as an example, additional inputs are required as we cannot revoke a person's identity in practise.

4. **Security Responsibility**: implication of domain information, "who is responsible for the holder on a certain matter", or "where to get started".

A profile certificate, PC_S, for S is two identifier fields, profile and security responsibility parameters, signed by the issuing $ProCA$'s private key. For instance, assume that the public key for a ProCA and for a company B is K^+_{ProCA} and K^+_B respectively. The different roles within company B are deployed to *profile* its employees, e.g. CompanyB.Marketing.Manager, CompanyB.Financial.Clerk, or CompanyB.IT.Engineer, and so on. A typical profile certificate PC_B for a company B will be,

Owner's Public Key	Issuer's Public key	Profile	Security Responsibility
K^+_B	K^+_{ProCA}	Role	CompanyB

"CompanyB" is used as the *security responsibility* in the PC_B. It is not an identity/name but an indication, "where to get started". It can be e.g. an on-line website, a mobile number, an email address, or a location based upon the context of different applications.

In terms of its function, the role used as the profile in company B's profile certificate differs from the one in the role-based access control systems [53, 106]. A role does not help verifiers to make an access control decision based upon which role company B's employees have. Instead, it intends to give the verifiers a clue what to expect in the employees' profile certificates. Specifically, a verifier will not expect to see an employee's profile certificate use "Main Building" as his/her profile when company B profiles its employees by role rather than geography. Thus, PC_{Bob} for the company B's marketing manager Bob will be

Owner's Public Key	Issuer's Public key	Profile	Security Responsibility
K^+_{Bob}	K^+_{ProCA}	CompanyB. Marketing. Manager	PC_B

Note that the profile for a local user is specific within a domain. Also, the domain's profile certificate is embedded into its local users' profile certificates.

For the profile certificate, note that there are two arguments here.

1. In the domain-oriented approach for access control over multiple domains, it is important that a malicious user cannot set up a false/bad domain at the beginning. Consequently, I argue that it is a fair assumption that $ProCA$ will be unlikely to *certify* an association between a malicious user and its own established false/bad domain. This can be realised mainly via some *social responsibilities* probably gained from other channels. For instance, to set up a legitimate company, we have to register with some business registration authorities; or a university

has to pass the higher education commission's assessment before it can recruit overseas students.

2. It also appeals that profile certificate authorities do not need to maintain any kind of certificate revocation mechanisms due to the impact from the domain-oriented approach for access control. The *social responsibilities* can also be applied here to revoke a domain's certificate, such as, a company has to withdraw its registration before its closure. The basic assumption is every player (i.e. domains, *ProCAs*) should be able to know if a domain is not here any more. For instance, a company "suicides" itself by broadcasting its closing down message. It is trivial to revoke a user's profile certificate because the final commit/abort decision comes from their local domains.

In conclusion, differing from the conventional semantics of *certificates*, the profile certificate is used here to give all verifiers an indication, *where to get started*. Purpose of the profile certificate is to improve performance, and the certificate needs to be "mostly right". This form of *certificate* is not used to make a final commit/abort decision. Instead, the access control decision is managed by the classical *authentication* and *authorisation* steps but with a new notion of delegation.

5.3 Delegation and Access Control over Multiple Domains

As described above in section 5.1, the domain-oriented approach for access control in the LoT framework is to reduce the (hard) access control across domains problem to two (easier) problems, *remote authorisation* between domains, and *localised authentication* in a single domain. As in figure 5.2 (on page 94), I assume Alice and university B as a resource domain and a users

domain respectively, Bob is a user from university B who intends to access the resource S owned by Alice.

5.3.1 Remote Authorisation - Delegation of Rights

Delegation is a natural consideration for making access control decisions [1]. Arguably, in terms of semantics, the conventional public key certificates are generally synonymous to most delegation systems. A public key certificate is deployed to *delegate* the belief on the association of a public key with other necessary information (e.g. access privileges). In the domain-oriented approach, the access control rights are managed by the resource domains, for instance Alice. Thus, when resources are shared between domains, the resource domain intends to *delegate* some access rights (on a particular resource) to the possible users' domains, very often with some necessary restrictions. Usually, Alice generates some forms of *access credentials* depending upon different delegation mechanisms that are deployed within her own domain. Then, she hands over the *access credentials* to the university B. In the multiple domain context, the requirements for defining such *access credentials* are:

- **Domain dependent**: credentials should be domain relative. It is important that different user domains should have different values of *access credentials* for the same access right on the same resource. For example, to delegate the "allow to make domestic faxes" right to university B and university C, Alice will generate two different bit patterns for these two universities. In this way, the compromise of one user domain does not help an attacker to gain any information of usable forms of access credentials for other user domains. Moreover, the delegation mechanism deployed in the resource domain needs to ensure that the collusion of some misbehaving user domains still cannot assist the attacker in terms of constructing a usable access credential.

- **Presentation restriction**: in most delegated access control systems, the possession of a delegated access credential is both necessary and sufficient to gain access, e.g. a secure capability. For the multiple domain context, on the contrary, I reason that the (direct) presentation of the domain-relative credentials is neither necessary nor sufficient for a user to access a certain resource. Actually, this presentation is extremely restricted after being issued. Alice will not be expected to grant the access request to Bob who naively submits the university B's access credential. Instead, Alice will regard university B to be compromised and revoke university B's access right because university B must explicitly delegate the credential to Bob.

Remote authorisation is done when access rights are delegated across domain boundaries from Alice to university B. Interestingly and in contrast with conventional approaches, university B does not delegate any conventional forms of access credentials to its local user Bob before the access. It may seem that re-delegation is being restricted. LoT does appreciate the importance of allowing free re-delegation but understands the semantical meaning of re-delegation from a different viewpoint. This is due to the domain-oriented nature for the access control mechanism in LoT.

As shown in figure 5.3, Alice does not care who Bob is as long as Bob's domain has the correct access credential and Bob has his domain's permission to use them. Hence, Alice will ask Bob to authenticate himself/herself to the correct university B.

5.3.2 Localised Authentication

Thus, the user *Bob* is essentially authenticated by his local domain, university B. We notice that the place to do authentication is different. It changes from the receipt's domain (i.e. the resource owner, Alice) to the sender's domain (i.e. the user domain, university B), as indicated in figure 5.4.

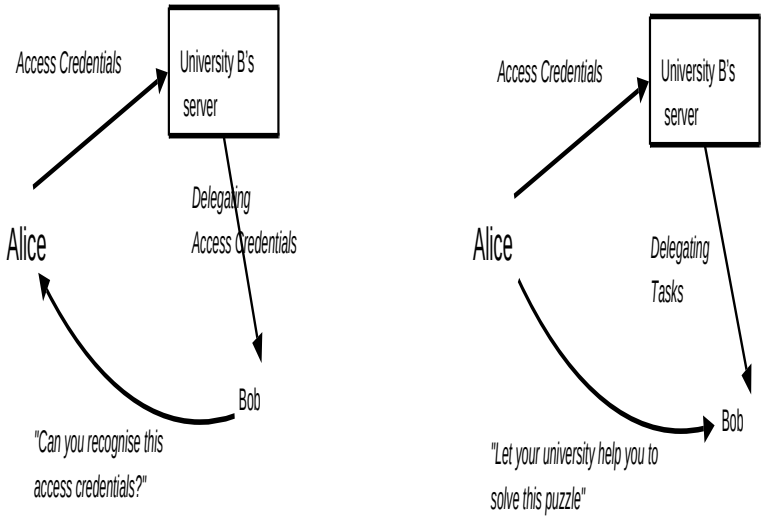

Figure 5.3: *The concept of the conventional approaches is briefly described in the left diagram. University B delegates the access credential (in some form) to its student Bob. Bob then submits this delegated access credential and expects Alice to be able to recognise them. As illustrated in the right diagram in our alternative scenario, university B does not delegate the access credentials in any form before Bob's request for access. Alice will ask Bob to authenticate himself to the correct university.*

The authentication step only occurs locally (within the users domain). A local *authentication channel* will be established between players and their local domains.

Local Authentication Channel: a communication channel is said to be an *authentication channel*, if it can sufficiently provide both the authenticity and confidentiality of any information exchanged between local users and their associated domains. This channel can be realised by many existing mechanisms, e.g. classic Kerberos like protocols [46, 122], policy-based trust management [16], or more pervasive security protocols [13, 45, 121], etc.

I do not intend to discuss those mechanisms in great detail in this dissertation. Instead, I encourage domains to choose a proper authentication mechanism freely and be responsible for their choice, in accordance with their own domain policies. The only requirement is that the *authentication channel*

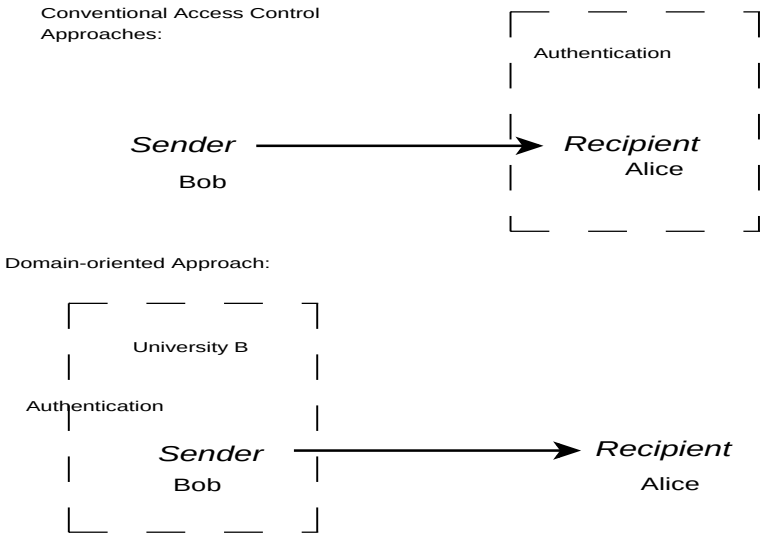

Figure 5.4: *For Alice, it is undoubtedly frustrating and difficult to authenticate someone from other domains. By allowing university B to authenticate Bob locally, Alice will not have to spend resources to understand the semantics of the authentication mechanism in a different domain, university B.*

established in the local domains will not be compromised in the future. Both users and local domains can be guaranteed that they are talking to the one whom they think they are.

For the localised authentication step, note that university B's server need *not* necessary be on-line all the time (I will explain this a little bit more in the next section, see page 110). From Alice's perspective, it is Bob's problem to authenticate himself/herself to not only university B, but the correct university B.

The basic idea of *localised authentication* has been hinted in the Identity-based Encryption cryptosystem (IBE). The IBE system does not require a chained or transitive trust relationship along the transmission path. Instead, trust is only established between end-users and a local trusted party, e.g. Private Key Generator (PKG) in the user's local domain. But nevertheless, to my best knowledge, this idea has not previously been explored significantly in the way that I will describe in this dissertation.

As a resource owner, Alice should *not* be forced to "contact" different possible universities every time their students come to request access. Hence, it is desired to have a mechanism connecting remote authorisation and localised authentication steps in the domain-oriented approach for access control over multiple domains. This can be realised by encryption-based access control.

5.4 Architecture for Encryption-based Access Control

Encryption-based access control (EBAC) is a relatively new concept, but it has been rapidly developed in pairing-based cryptosystems [18, 61, 117]. From the perspective of encryption, the encryption-based access control sketched here is analogous to some existing Identity-based encryption (IBE) schemes [2, 51, 63]. For instance, in figure 5.2 on page 94, it lets Alice encrypt a message in a way that *Bob* can only decrypt with the necessary assistance from some security services in his/her own domain (university B). As a consequence, it is *Bob's* responsibility to convince his (or her) local authority to issue the corresponding decryption keys (precisely a partial decryption key). Moreover, the partial decryption key can only be generated and supplied by *correct* university B.

From my perspective, pairing-based IBE is not suitable for the multiple domain context. In those systems, every domain's public key has to be available, at least at the time when a user requests the access. If the authenticity for the domain's public key is *certified* by some external certificates authorities, the certificate revocation and trust transitivity problems re-emerge.

The basic construction for encryption-based access control proposed in this dissertation is converted from Goldreich *et al.*'s self-delegation scheme [54]. In Goldreich *et al.*'s original system, their purpose is to delegate certain rights from a user to a user himself without risking the compromise of his long-term private/public key pair (primary public/private key). Accordingly, secondary

key pairs (sk_ℓ, pk_ℓ) are created by the user. They can only be validated with a validation tag (val_ℓ) based upon a certain limitation (the limitation index ℓ). Given a triple (sk_ℓ, val_ℓ, pk_ℓ), the player is able to convince a verifier that a certain public key pk_ℓ can be used on behalf of the primary key (given the limitation index ℓ, a primary public key and the necessary system set up parameters). I take advantage of the fact that a private/public key pair can also be applied to do decryption/encryption operations. Assume that PK^* is the full encryption key and SK^* is the full decryption key.

If the full decryption key SK^* can be computed by university B, restricting misbehaving domains and users in EBAC will be hard. Two specific problems are considered here,

1. The denial-of-service attack from university B. University B may maliciously do a denial-of-service attack and claim that the access comes from one of its students.

2. Corruption from Bob. If Bob decides to collude with another student Moriarty from very beginning, he can just simply "hand over" SK^* to Moriarty after the key is issued from university B.

To address those two threats, thus, the technique used in EBAC is to separate SK^* into two parts, partial decryption key (PDK) and R generated by the user's domain and the user itself, respectively.

The overview for encryption-based access control described above is illustrated in figure 5.5. Those steps will be describe in detail in next chapter. Briefly speaking, Alice delegates a domain-based access credential (AC_B) to university B (i.e. its domain server). When a student Bob from university B requests to access a particular resource owned by Alice, she will give Bob an access "token" based upon Bob's access request. This "token" can only be converted to a usable access capability if the correct university B issues a

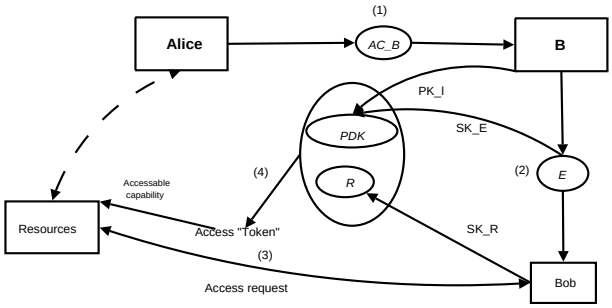

Figure 5.5: *To access Alice's resources, a student Bob has to retrieve the a partial decrytion key (PDK) from university B.*

PDK at the time of Bob's access. Thus, it is Bob's job to convince university B that he is entitled to access this particular resource. As far as Alice is concerned, she does not care how Bob authenticates himself to his own university. She is happy to grant Bob's access once the conversion is completed successfully [3].

For PDK generated by user domains: PDK can be provided by university B. It is based upon the access credential (AC_B) delegated from Alice to university B during the *remote authorisation* step. As described above, however, the presence of AC_B is extremely restricted. Therefore, the requirements for defining PDK are to be,

- Session-dependent: for every access request from its students, the PDK issued from university B will be different. This is mainly to avoid *Bob*'s dishonest behaviour or the replay attack. In addition, similar to the case of environmental key generation [97], university B may restrict *Bob* to allow access only if certain classes of environmental conditions are true, for instance, on a certain date, at a certain time, and so on. University B can take those local restrictions, environmental requirements and *Bob's* personal information, and denote them by access index ϵ.

[3] If *audit trail* is one of the concerns for some applications, another "audit token" can be provided by university B (and embedded in the access token). Alice cannot decrypt this audit token, nor re-produce it. However, if something is going wrong, Alice can present the audit token issued by university B simply for the purpose of the audit trail.

- The second requirement is quite straightforward. Given any collections of issued $PDKs$ from a set of students in university B, an attacker cannot compute the value of AC_B.

In EBAC, SK_ϵ is added to generate a PDK. It is the secret part of the key pair (SK_ϵ, PK_ϵ) generated by university B after inputting the access index ϵ. Moreover, the use of SK_ϵ is to guarantee that the AC_B will be kept secret to university B (the users domains). Also, this key pair (SK_ϵ, PK_ϵ) is called the *Endorsement Pair* in EBAC.

For R generated by users: the generation of R is based upon the idea of *delegation of responsibility* [36] (see the brief review in section 2.3.2 on page 40). In the domain-oriented access control approach in LoT, users are *organisationally affiliated* with their local domains. EBAC lets Bob contribute a secret component R. R is the secret part of the key pair (SK_r, PK_r) generated by Bob. It plays two important roles here. First of all, the existence of R prevents a misbehaving university B to masquerade as *Bob*. Secondly and more importantly, if *Bob* decides to collude with another student Moriarty from the very beginning, *Bob* has to give SK_r to Moriarty. This will force Bob to compromise his personal secret to an attacker, in order to breach security. We can consider for example that this (SK_r, PK_r) pair is associated with some form of digital cash [7, 23, 100, 124] for Bob. Thus, Bob is not willing to give up SK_r by any means. The key pair (SK_r, PK_r) is called *Responsibility Pair* in EBAC.

For AC_B generated by resource owners: as discussed in section 5.3.1 (see page 101), AC_B is required to be *domain-dependent*. Thus, Alice generates a master secret SK firstly in her own domain and computes a pair (SK_ℓ,PK_ℓ) for university B. This pair is called the *access rights pair*, where PK_ℓ is AC_B. From the requirements of being *domain-dependent*, we know,

- Given any single value from PK_ℓ, it is (computationally) infeasible to

compute the corresponding SK_ℓ.

- Given a set of values from a set of PK_ℓ, it is still (computationally) infeasible to compute any SK_ℓ or the master secret value SK.

The construction of the access rights pair will be highly dependent on the mechanism that Alice chooses to use to delegate access rights, e.g. a secure capability, a public key pair, and so forth.

Encryption/Decryption: now, we will have,

- The Partial Decryption Key (PDK)

$$PDK = AC_B \cup SK_\epsilon = PK_\ell \cup SK_\epsilon, \textit{ (1)}$$

- A full encryption/decryption key pair,

$$\text{Encryption key: } PK^* = SK_\ell \cup PK_r \cup PK_\epsilon, \textit{ (2)}$$
$$\text{Decryption key: } SK^* = PDK \cup SK_r = PK_\ell \cup SK_\epsilon \cup SK_r, \textit{ (3)}$$

- Two algorithms, *Encrypt*{} and *Decrypt*{}.

  1. *Encrypt*{}: this algorithm is used to encrypt a message M under the encryption key, PK^*, after inputting SK_ℓ, PK_r, PK_ϵ.
  2. *Decrypt*{}: correspondingly, it is called by Bob to recover the message M by taking the full decryption key, SK^*, after inputting PDK and SK_r.

Note that "∪' here is only a symbol. The mathematical meaning varies depending on the underlying cryptographic algorithm chosen in the implementation. More practical approaches will be given in next chapter.

Thus, for EBAC,

1. By having a secret input R from a user, e.g. a student Bob, two threats, the denial-of-service attack from university B and the corruption from Bob can be addressed as discussed above.

2. By having a secret input SK_ϵ in PDK from a user domain, e.g. university B, university B does not have to be on-line all the time. University B can send the PDK for Bob to a directory for example. Bob only needs to get this PDK from the directory.

Generally speaking, revocation works closely with access control, the delegation semantics in particular. Revocation discussed in access control context is mainly subject to the various delegation relationship among resources servers domains, users domains and associated local users. We cannot discuss *delegation* without having *revocation* in mind.

5.5 Revocation

From a security perspective, *revocation* is a way to prevent unauthorised access resulting from the use of some invalidated statements. But meanwhile revocation is one of the main difficulties for many security systems. A notorious case is the (public key) certificate revocation problem that has been witnessed in many PKI-based systems. Conventionally, the revocation problem has been dealt with separately and some ad-hoc mechanisms have been provided for managing revocation information [59]. However, there are two problems of doing so,

1. From the infrastructure's perspective, users will have to understand the semantics of revocation separately. Unfortunately, it is difficult to believe that users will understand and check the detached revocation mechanisms at all. For most conventional revocation schemes, it is up to one communication participant to check if the information is revoked. For simplicity or performance reason, users may be more likely to choose not to use additional revocation mechanisms, particularly if this request is *forced* by the infrastructure. In pervasive environments, it is infeasible and unnecessary to require a user from one domain to understand the

semantics of the revocation mechanisms in other domains. Moreover, if the infrastructure pushes applications to check revocation by default, each domain has to make sure their own security policies satisfy the infrastructure's requirements and assumptions. This is not what we want, particularly in the multiple domain context.

2. For the semantics of revocation itself, a more serious problem is the *timeliness* of revocation information. In short, revocation requires users to obtain necessarily real-time revocation information. For the conventional approaches, both *pull* and *push* by (revocation information) providers suffer from this problem. This *timeliness* concern is more significant for pervasive environments. As pervasive communications usually involve multiple domains, a domain may not always be aware of any security policy changes in other domains. Again, the conventional approach to this is to let the infrastructure play a crucial role. The infrastructure is responsible for collecting the relevant revocation information from different domains. Then, the users from the other domains can check with the infrastructure, if they need. It is doubtful to have in place such an infrastructure always accessible by remote domains in pervasive environments. Also, the extra cost to manage revocation will be significant because most pervasive applications may be taking place on a purely temporary basis.

Here are the two main issues I understand for revocation in EBAC in the LoT framework.

- Revoking local users' access request: this is the primary issue in this dissertation as far as *revocation* is concerned. The domain-oriented approach, particularly the EBAC method, brings many advantages in terms of addressing the revocation problem. Under some circumstances, university B may want to revoke a student Bob's access rights on accessing

the shared resources owned by Alice. It may be the long-term case, e.g., Bob is not a Ph.D student any more. Or, it may be for a temporary invalidation, e.g. Bob takes one day's leave from the university. For both cases, providing the real-time revocation information is crucial but unfortunately most revocation mechanisms fail to do so efficiently.

In contrast, the EBAC method deployed in the domain-oriented approach manifests the efficiency of revoking the users' access request. It is considered to be part of normal transaction. As shown in figure 5.5 (on page 107), Bob cannot generate a full decryption key without acquiring a *current* PDK from university B. Moreover, Bob is not able to compute the correct PDK on his/her own. Consequently, the PDK can be considered as the *revocation factor* created by university B. To revoke Bob's access request, the only thing university B needs to do is to simply stop issuing the PDK, when Bob requests the necessary crypto key materials in the real-time interactions. Once B drops its pointer to the PDK, the access from Bob immediately stops forwarding. All the access statements Bob has become useless.

- Revoking delegated access rights across domains: In EBAC, access rights are managed by resource servers domains. The rights are delegated only at the domain level, from a resource servers domain (Alice) to a users domain (university B). Alice may not want any students from university B to access a certain resource any more, for instance, the contract expires. For the domain-oriented approach, she just needs to revoke the proper access rights for university B. Since revoking access rights takes place within her own domain, Alice can maintain a domain-based access control list *locally* [4]. If she wants to revoke *a domain's* access

[4]Having a *local* access control list is not a step backward. Like Karger's S-CAP [69], an access control list is adequate here as long as it can be kept *local* to Alice's own domain.

rights, she deletes this domain's profile certificate from the *domain access control list*. To compose a user's profile certificate, the domain's profile certificate has to be included. As a result, if a requester's PC contains a revoked domain's information (more precisely, the domain's profile certificate), Alice will abort the communication.

Advantages: the domain-oriented approach for access control over multiple domains, particularly the EBAC method, has the following properties in terms of revocation.

- *Integration*: instead of being treated separately, revocation is (partially) integrated into the normal transaction. The principle is rather simple. University B releases the PDK if its local students' access requests are still validated from its own perspective. Otherwise, university B stops the issue of the PDK. This scheme essentially reduces the cost that is used to establish additional revocation mechanisms.

- *Immediateness*: Alice will know Bob's access statement has been revoked immediately, if Bob cannot successfully decrypt the message encrypted by Alice. Revocation occurs in real-time.

- *Seletiveness*: under some circumstance, university B may just want to revoke Bob's access temporarily, e.g. for one interaction or one day. Using the EBAC method, university B does not need to ("physically") revoke anything. Instead, university B stops issuing PDK to Bob for this instance. When Bob resumes his job next day, domain B starts to issue the PDK again.

- *Revocation Transitivity*: domains' profile certificates are cascaded into their own users' ones. Thus, once university B's certain access right has been revoked, access requests from any students from university B will be denied automatically. Again, a player's profile certificate does

not have to be *accurate* because the finally decision will go back to the player's local domain.

Thus, the PDK is like a freshest certificate, however,

- A player, Bob, has to have it to access a resource.
- It is domain B's problem if issuing a correct PDK to correct Bob goes wrong.

5.6 Conclusions

The entire design of the Encryption-based access control scheme is motivated by the domain-oriented viewpoint in the LoT framework. It is based upon the concept of localising the trust. More specifically,in LoT the commit/abort security decision is ultimately managed by the local domain, rather than external authorities. This is an efficient approach because a user does not need to understand the semantics of security mechanisms in other domains. Moreover, the EBAC scheme effectively solves the inherent revocation bottleneck in most delegation focused access control systems. The most difficult aspect in terms of the revocation problem is addressed by treating revocation as part of the normal transaction in the EBAC scheme.

The domain-oriented EBAC scheme is the central stone for LoT. In the next chapter, I will investigate two access control scenarios, *stranger access* and *guest access*. Also, I will describe the details of implementing the EBAC scheme enabling them in pervasive environments.

Chapter 6

Semi-Trusted Agent Mediated Protocols for Access Control

Two different scenarios will be examined in this chapter, as highlighted in section 3.5 (see page 67). A player who intends to access some shared resources from other domains may be a stranger to the resource domain. Or, a player can be from a domain who is unknown to the resource domain, however, the player is *friendly* (most likely on a purely temporary basis) to some of local players from the resource domain. Those two scenarios are referred to, *strangers access from friendly domains*, and *guests access* respectively. My attention will be focused upon the *strangers access from friendly domains* because arguably the guests access scenario is indeed a case of access control for a single domain (I will explain this later on in section 6.5).

This chapter begins with an overview of the Semi-Trusted Agent Mediated Protocol for achieving access control in pervasive environments. Then, I illustrate an example threat model for strangers access from friendly domains in section 6.2. Section 6.3 describes a Dual Capabilities Model, which is used for domain-based access control. Detailed protocol implementation for achieving localised authentication in encryption-based access control is proposed in section 6.4. This chapter ends with a brief analysis of the guest access scenario.

6.1 The Overview of the Semi-Trusted Agent Mediated Protocol (STAMP)

STAMP in the LoT framework is designed to provide necessary access control in the multiple domain context, and has been optimised to work in pervasive environment. Essentially, it has three primary components: resources, agents and (domain) servers. A *resource* is any type of resources or services within a domain that are potentially accessible by users, or shared with other domains. It could be either hardware, e.g. a DVD player, a fax machine, a wireless camera, or some software programs running on some devices.

The architecture for STAMP is based upon *domains*. Each domain is controlled by a *domain server* whose responsibility is to manage its *local* users and resources. It allows users and services/resources to register, and provide appropriate *profile certificates* [1] to them, for instance. In addition, two types of agents have been employed by STAMP, *capability agents* and *semi-trusted agents* (or STAs).

- Capability agents: the *capability agent* maintains a *User Domain-level Access Control List* for any shared services/resources owned by the domain. When a user requesting the access (on a particular resource) is from other (user) domains, the capability agent will firstly check if the domain this requester belongs to has the appropriate access right. Moreover, the capability agent is responsible for delegating the access rights between domains during the remote authorisation process (that is, cross-domain delegation phase). The details are described in section 6.3.

- STAs: STAMP introduces semi-trusted agents to extend the domain-oriented approach for access control over multiple domains. Those agents

[1] As indicated in section 5.2, the profile certificates for both domains and their affiliated users ought to be issued by a local profile certificate authority.

are *trusted* only by their own associated domain servers. More specifically, the communication channels between domain servers and those agents are assumed to be secure, i.e. authenticated, non-repudiated and confidential. However, *neither* local users *nor* domain servers from the other domains need to *trust* those agents at all. For instance, a player may not have accessed a particular local resource before and therefore has not talked to a particular STA at all. Thus, those agents are considered to be *semi-trusted*. They can be in different forms with respect to the nature of multiple-domain applications, such as hardware devices, software running at a personal device, or a website, and so forth.

Figure 6.1 shows an overview of the STAMP architecture from one domain viewpoint. Note that multiple semi-trusted agents may exist for one domain.

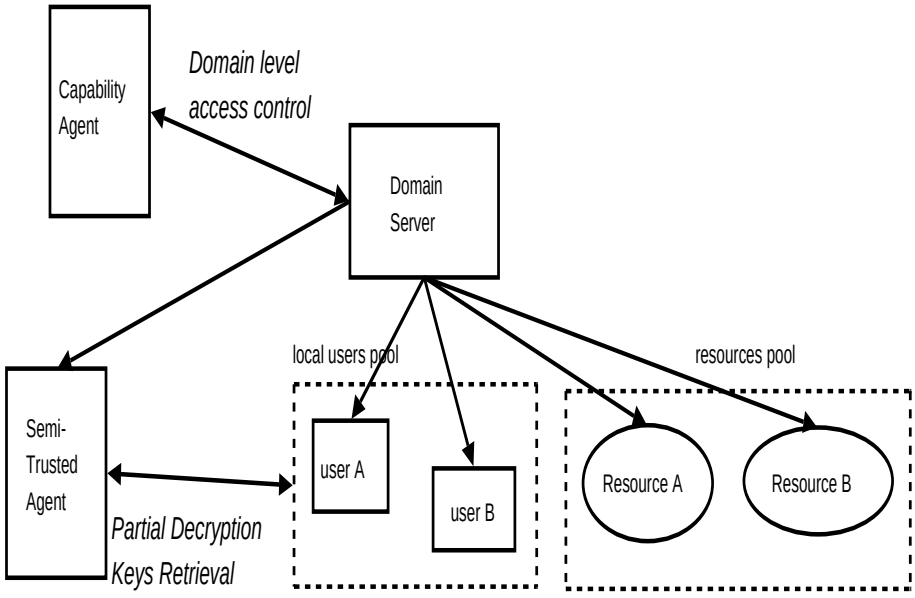

Figure 6.1: STAMP Overview
The overview of STAMP

Roughly speaking, the STAs' major purpose is for users to retrieve partial decryption keys (PDK) that are deployed in the encryption based access control method. As a consequence, *localised authentication* is accomplished.

Semantically, STAMP follows the description of the domain-based approach for access control over multiple domains. Users' own domains are responsible to make the final commit/abort decision for their local users via supplying the necessary partial decryption key (PDK). The success of the user retrieving the correct PDK will convince the remote resource owner that the access requester has the proper authorisation to access some *shared* resources across domain boundaries.

Semi-Trusted Agents: The semi-trusted agents have a vital role in the STAMP architecture. There are three major reasons to have such agents here.

1. An advantage of the domain-oriented approach for access control is that the domain server controls the necessary PDK for its local users. However, it also means that the domain server has to be on-line all the time for issuing the PDK, and more important dealing with real-time revocation. By employing *local* agents, domain servers can be kept off-line after sending PDKs to agents at the beginning of every period of time (let us say every day). Domain servers only need to be on-line if they would like to change their local users' states (e.g. revoking or re-validating).

2. The STAs from one domain may be geographically located in different places. However, they are based upon domains and therefore *local* to their own domain servers. In other words, parts of them act like local agents. Thus, it is fair to assume that the secure channel between STAs and their associated domain servers is always available. Generally, to jam this channel is both difficult and expensive in practice. More importantly, the responsibility to keep the channel open rests entirely within one domain.

3. Moreover, having an agent brings the potential benefit of keeping the resource devices themselves simple. In pervasive environments, those

resources devices can be wearable gadgets, DVD players, kitchen appliances or any lightweight devices. If the resource device has limited computational power, they can communicate with nearby agent(s), allowing agent(s) to run some complicated tasks for them [2].

6.2 Strangers Access from *Friendly* Domains

6.2.1 Motivating Threat Model - Public Meeting 2

I again focus upon a simple but interesting (in the security protocol sense) pervasive application, the public meeting. It has already been described in section 4.1.2. The company A's marketing manager Alice meets company B's marketing manager Bob for the first time in a public conference room. She would like to securely transfer a private project plan m from DRD_{alice} to Bob's hand-held device DRD_{bob} this time. I assume that the context of human trust between Alice and Bob is missing in this scenario, for instance, Alice does not know *Bob* in person. In other words, *Bob* is a stranger to Alice although company B is known to Alice. Thus, Alice has to be convinced that she is talking to the right *Bob* as well as talking to the right device.

Considering the basic approach in chapter 5, the computations might be the same. The semantics, however, are totally different with respect to the change of context in this threat model. Previously, the weak secret k is generated randomly and agreed by Alice and Bob. Now, it has to be shared between two *friendly* domains, company A and company B, for this meeting's purpose first. Both Alice and *Bob* are delegated this secret value from their own company. It is also necessary that Alice and *Bob* load this pre-shared secret into their devices respectively. They have no chance to observe each other's inputs during the meeting. If this secret is leaked to an insider attacker (e.g.

[2] A secure communication channel between a device and a agent in a either wired or wireless network can be established by using many existing security protocols that have been widely developed in conventional environments. The discussion on this issue is out of the scope of this dissertation.

a bad guy from company B) somehow, he (or she) can masquerade as *Bob* without Alice's awareness. Hence, the security is being provided in the wrong place from the very beginning, as Alice talks to the wrong *Bob*.

It does not get better if security protocols require *Bob* to show some proof (conventionally somehow relating to IDs again), e.g. traditional personal ID card issued by company B, or a signed letter from the company B [3]. Alice may not be able to recognise those proofs issued by the company B because she is from a different company A. Even if she may, it is still difficult and frustrating for her to be aware of any policy changes which have occurred in the company B.

Thus, the main threat I try to address in this example is the *confinement* problem associated with the stranger access from a friendly domain, in contrast to the *man-in-the-middle* attack in the public meeting 1 scenario.

6.2.2 Confining an Insider

I (deliberately) associate the stranger access scenario with the classic *confinement* issue here. The *confinement* problem is defined by Lampson [76] in the early 70s. It is to determine whether a series of operations will pass information to an unauthorised process or not. Access control policy is an important security policy to address the confinement problem. It is fairly straightforward to prevent an outsider from gaining access to unauthorised resources for most access control systems. However, the problem is the unsuspecting insiders. Particularly, in the case of stranger access (from *friendly domains*), Alice may not be aware of the *semantic* difference between a stranger authorised by the stranger's own domain and an inside attacker who is from the same domain as the stranger.

Loosely speaking, in this dissertation,

[3] We can think of this signed letter as the form of traditional public key certificates in the cyber-world.

Confinement: to confine potential damage from an insider within the local domain. We can take the public meeting as an example, Alice should be informed to abort at time of interaction, if a usable access credential has been leaked to another unauthorised user in company B.

Moreover, *confinement* is a mechanism to restrict an insider's wrongdoing. We do not expect to see that an insider Moriarty could masquerade as Bob by, for instance, stealing the secret credential on Bob's personal device. On the other hand, if Bob is misbehaving, he should be the one who suffers the most.

STAMP can be deployed here to confine an insider. The delegation of rights is achieved by the remote authorisation step through *capability agents*, as discussed in the previous section. Consider spontaneous and dynamic pervasive interactions, we need a flexible mechanism to delegate access rights between domains in STAMP. This naturally leads us to a capability-based access control method.

6.3 Dual Capabilities Model (DuCaM)

Dual capabilities deployed in DuCaM are similar to the idea in I-CAP [55] and split capability proposed in [72]. The dual capabilities heavily borrow Gong's novel idea of having internal and external capabilities. However, the basic system infrastructure for DuCaM is built from the domain-oriented approach for access control (over multiple domains), which is completely different to I-CAP. As pointed out in section 2.2.3 (see page 35), an additional authentication mechanism is required in the I-CAP system to check the user's ID at time of using the external capability. In DuCaM, restricting the presentation of (external) capabilities is utilised to secure the capabilities instead of requiring a reliable authentication mechanism across domains. In DuCam,

authentication occurs only locally within the user domain. From this significant semantic difference, capabilities used in DuCaM are not simply access control oriented *tokens* only. In fact, their primary task is to let (remote) users authenticate themselves to their own domains, and more importantly to the *correct* domains.

Going back to the public meeting case (on page 119), assume Alice (the resource owner), company B (a user domain) and Bob (a stranger from a user domain) have their issued profile certificate, PC_A, PC_B and PC_{bob} respectively. Note that some notions used below have been explained in section 5.4.

I only consider one direction of trust, in which Alice would like to make sure Bob is the right stranger from company B, purely for simplicity. If mutual trust is required (e.g. the meeting for two secret societies), the same protocol will be used but changing the direction.

Dual capabilities: the *capability agent* managed by Alice generates dual capabilities for some possible organisations, for instance company B, on a shared resource, the project document ob_m in the public meeting case.

- *Internal Capability*: this keeps the same form as the one in I-CAP. The capability agent generates a random secret r, for some resources owned by Alice, e.g. the document ob_m.

  $$IntCap = (ob_m, r),$$

  where, r is only known to Alice's capability agent as a master secret.

- *External Capability*: company B initiates a service request by sending the profile certificate (PC_B) to Alice. Alice checks her own local access control policy, and inform her capability agent to delegate appropriate access right to company B. The capability agent updates the domain-level access control list (on ob_m) by inserting an new entry of PC_B, then produces an external capability for company B,

$$(ExtCap_B),$$

where,

$$ExtCap_B = h\{ob_m, r, AccessRights, PC_B\}$$

$h\{\}$ is an one-way hash function. Eventually, the external capability will be handed over to university B via Alice.

The basic diagram is illustrated in the figure 6.2,

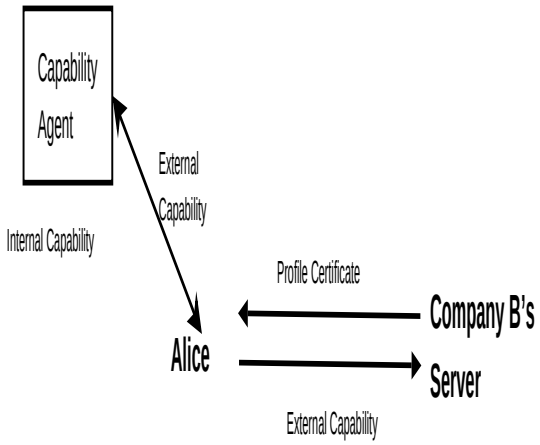

Figure 6.2: *The resource owner Alice's capability agent generates a domain-dependent credential, external capability, for a user domain company B.*

We can think of the internal capability as a master secret SK, and the *access rights pair* (SK_ℓ, PK_ℓ) introduced in previous chapter are,

$$SK_\ell = ob_m, r, AccessRights, PC_B,$$
$$PK_\ell = ExtCap_B = h\{SK_\ell\},$$

The possession of an external capability in STAMP is not sufficient to gain access to a certain object. The main purpose of having a domain-specific external capability is not for the (conventional) access control purpose only. Essentially, it is used for the localised authentication step. Within company B, the external capability will not be delegated to any employees in any form

before the access. Instead, it will be stored in a safer place (than carrying around), e.g. the company B's server, or some other places that company B's server has a stable connection with.

Access process: when Bob requests access on obj_m, Alice will respond to Bob with a form of "token" (using the encryption-based access control method). Essentially, whether Bob can use the information encrypted in this way by Alice is not Alice's concern at all. It is Bob's job to convert the "token" to a usable material. Moreover, Alice knows the right company B has to get involved to accomplish this conversion successfully. Hence, if this can be done by Bob, it is good enough to convince Alice that the person she intends to talk to is from the correct company B for the current meeting's purpose. In other words, from Alice's perspective, authentication takes care itself.

6.4 Discrete-Logarithm based Scheme for Localised Authentication

In this section, I will describe a practical implementation to achieve localised authentication in the context of EBAC. The underlying cryptographic scheme of this example approach is based on discrete logarithms. More precisely, I assume a generator g, large prime modulus $q = 2p+1$ for prime p, and one-way hashing function h are publicly known. Also, random numbers generated from the underlying algorithm must be strong (i.e. long enough to be invulnerable to exhaustive search) as well as hard to predict.

For a chosen decryption/private key SK, the corresponding encryption key (public key) is $PK = g^{sk} \bmod q$. A typical DL-based Encryption/Decryption example is ElGamal scheme [42]. Roughly speaking, for example, if C wishes to encrypt a message m for D with D's public key, g^d (where d is the corresponding private key for D), C should do the following,

- Select a random integer k, where $1 \leq k \leq q-2$.

- Compute $C_1 = g^k \bmod q$ and $C_2 = m \times (g^d)^k \bmod q$.

C sends C_1, C_2 to D. D can recover the m from C_1, C_2 using the private key d by calculating ${C_1}^{-d} \times C_2$. The $Encrypt\{\}$ and $Decrypt\{\}$ in this section will use these steps.

Some procedures described later stem from the approach in section 4.4.2 (see page 86). They are mainly used to establish a two-channel protocol for authenticating the right device. Briefly, the two channels introduced previously, one the RF channel and the other an out-of-band channel (OoB), will be deployed here as well. I assume that DRD_{alice} is a device for Alice, B is a server for company B and DRD_{bob} is a personal device for Bob. STA_B is the *semi-trusted agent* for company B, for instance, a website accessible from DRD_{bob}. Again, the channel between company B's server (B) and this website (STA_B) is secure, and it is difficult and expensive to jam such a secure channel in practice.

The Profile Certificates for company B and Bob are PC_B and PC_{Bob} respectively, where,

$$PC_B \text{ is } (K_B^+,\ K_{ProCA}^+,\ Role,\ CompanyB) \text{ signed by } ProCA,$$

and,

$$PC_{Bob} \text{ is } (K_{Bob}^+,\ K_{ProCA}^+,\ CompanyB.Marketing.Manager,\ PC_B) \text{ signed by } ProCA.$$

The basic protocol is highlighted in figure 6.3, which follows the overall infrastructure of encryption-based access control introduced previously (see figure 5.5 on page 107).

Responsibility and Endorsement Pair: company B delegates the task, "access a shared project document owned by Alice", to its employee Bob. STAMP recalls the encryption-based access control method here and requires

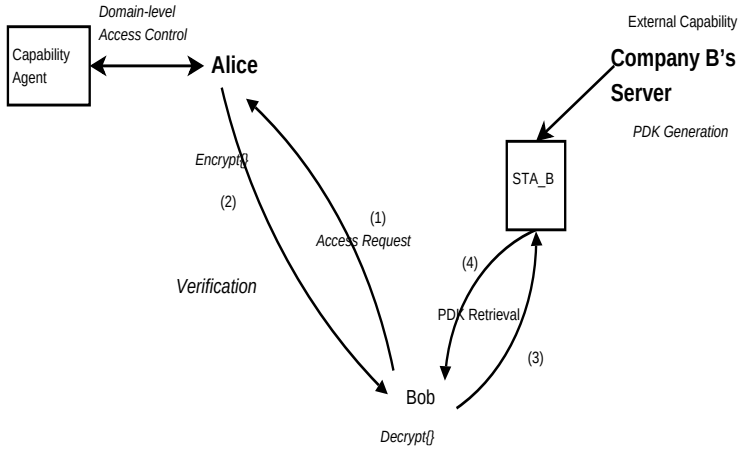

Figure 6.3: *Localised Authentication in STAMP*

the generation of the *responsibility pair* and the *endorsement pair* for a user domain, i.e. company B.

Bob (DRD_{bob}) generates a *responsibility pair*, (SK_r, PK_r) from the underlying discrete algorithm, where,

$$PK_r = g^{SK_r} \bmod q,$$

In the meanwhile, a local authenticated channel (LAC) is established as well between the company's server B and DRD_{bob}. The purpose of LAC is for localised authentication. DRD_{bob} submits PK_r to B,

$$DRD_{bob} \rightarrow_{LAC} B\text{: } PK_r,$$

Company B inputs a certain local restriction index (ϵ) along with PK_r to compute a *endorsement pair* (SK_ϵ, PK_ϵ). That is,

$$PK_\epsilon = g^{SK_\epsilon} \bmod q.$$

Then, the delegation of task, accessing Ob_m, is issued from company B to Bob,

$$B \rightarrow_{LAC} DRD_{bob}\text{: } PK_\epsilon, Ob_m, AccessRequest, PK_r,$$

Access Request: *Bob* meets Alice in a public conference room. DRD_{bob} generates a random number b and a weak nonce n_b, then requests the access on the project plan (Ob_m),

$$DRD_{bob} \rightarrow_{RF} DRD_{alice}\text{: } PK_r,\ PK_\epsilon,\ Ob_m,\ AccessRequest,\ PC_{Bob},$$
$$(g^b + n_b \bmod q),$$

At the same time, DRD_{bob} sends the n_b to A via the infrared link,

$$DRD_{bob} \rightarrow_{OoB} DRD_{alice}\text{: } n_b,$$

Encryption Phase:

Alice obtains the domain information PC_B from PC_{Bob} and sends it to her capability agent. The capability agent will have to look up the (local) domain-level access control list to make sure company B's access rights on Ob_m are still validated and appropriate. If so, the capability agent re-constructs a permissible [4] external capability from the internal capability (Ob_m, r) and access request committed by DRD_{Bob},

$${ExtCap_B}^\ell = h\{Obj_m, r, AccessRequest, PC_B\},$$

Alice (DRD_{alice}) receives ${ExtCap_B}^\ell$ from the capability agent. Then, she also generates a random number a and picks a weak secret n_a. Alice runs the $Encrypt\{\}$ algorithm by inputting the value of $g^a + n_a \bmod q$,

$$Encrypt\{\}\text{: } C = E_{PK^*}[g^a + n_a \bmod q],$$

where the encryption key (recall the fomula *(2)* on page 109) ,

$$PK^* = SK_\ell \cup PK_r \cup PK_\epsilon = (PK_\epsilon)^{h\{SK_\ell\}} \cdot PK_r =$$
$$(g^{SK_\epsilon})^{{ExtCap_B}^\ell} \cdot g^{SK_r} \bmod q,$$

Alice sends this encrypted information to *Bob*.

$$DRD_{alice} \rightarrow_{RF} DRD_{Bob}\text{: } C,$$

Also, DRD_{alice} sends n_a via the second channel,

$$DRD_{alice} \rightarrow_{OoB} DRD_{bob}\text{: } n_a,$$

[4] Bob may request some access rights to which he is not entitled.

If *Bob* is who he claims to be, he/she is now able to decrypt this message.

Furthermore, Alice computes,

$$g^{2ab} \bmod q = (s|n_1|n_2)$$

where s is the (potential) session key, and n_1 and n_2 are nonces.

Partial Decryption Key Generation Phase: After authorising Bob to access the document ob_m, company B's server B computes appropriate PDK_B (recall fomula *(1)* on page 109),

$$PDK_B = PK_\ell \cup SK_\epsilon = ExtCap_B \cdot SK_\epsilon (\mathrm{mod}(q-1)),$$

Then, B updates the STA_B with this appropriate PDK_B for the user Bob.

Bob retrieves PDK_B from STA_B via the established local authentication channel. Assume that the link to the website (STA_B) is also embedded in the crypto-module of DRD_{Bob}. Thus, the input (the link to the STA_B) and the output (the PDK_B) may be observed by the attacker if a spyware is successfully installed in DRD_B. However, the attacker cannot change the values.

$$STA_B \rightarrow_{LAC} DRD_{bob}\colon PDK_B,$$

Decryption Phase:

DRD_{bob} runs the $Decrypt\{\}$ algorithm by taking the PDK_B and the secret SK_r. The full decryption key is (recall fomula *(3)* on page 109),

$$SK^* = PDK_B + SK_r = ExtCap_B \cdot SK_\epsilon + SK_r (\mathrm{mod}(q-1)),$$

if *Bob* is the right *stranger* from a friendly company B, we should have

$$ExtCap_B{}^\ell = ExtCap_B,$$

Thus, DRD_{bob} would be able to get the correct decryption key (SK^*) and further to decrypt C. DRD_{bob} computes $g^{2ab} \bmod q = (s|n_1|n_2)$ as well.

Verification Phase:

Both DRD_{alice} and DRD_{bob} do the following step,

- $DRD_{alice} \rightarrow_{RF} DRD_{bob}$: n_1,
- $DRD_{bob} \rightarrow_{RF} DRD_{alice}$: n_2,

If the crypto-module from DRD_A finds they match, then Alice considers the current access requester is from the correct company B for this current meeting purpose.

Thus, Bob is authenticated by his local domain, company B. If an inside attacker Moriarty comes to meet Alice, he has to compromise the localised authentication mechanism in company B. Otherwise, he cannot retrieve the correct PDK from STA_B. From this, we can see that STAMP provides an efficient solution to enforce the *confinement* for access control over multiple domains.

6.5 The *Guest Access* Scenario

The second scenario considered here is friendly access from strange domains. Among numerous pervasive applications, increasingly, human users intend to access some resources that are provided by other domains, e.g. organisations, schools or other human users, on a purely temporary basis. For instance, an external consultant tries to access a client's Smart Spaces [68, 21] services, a guest wants to use the host's TV or refrigerator, or a visiting researcher would like to connect to the university B's secure wireless network for a short period of time during his visit, and so forth. Those examples indicate the necessity of requiring *guest access* in pervasive environments.

The guest access scenario can easily be mis-regarded as a case of access control over multiple domains, as the guests are from other foreign domains. However, it is really a single domain case from the domain definition in LoT. The intention of discussing this single domain example here is to enhance the correct understanding of *domain* employed throughout this research. It does

not, however, exploit the possible protocols to address the security issues in the guest access scenario in detail.

I will explain this argument in detail in the rest of this section. First of all, let us understand the characteristics of guest access.

6.5.1 The Definition of *Guests*

The scenario of *guest access* is different from the problem of talking to *strangers* (which has been discussed early in this chapter), as explained in [66]. An essential characteristic of *guest access* is that guests can only access resources with the necessary permission from authorised users, e.g. a family member from the host, the local office employee. A guest has to get the host's permission, in order to play a video clip stored in his/her DRD on the host's TV. However, the guest may not be permitted to access the host PC at all depending upon the host's preference. Thus, three facts about *guests* are described as follows,

- *Guests are not strangers.* Guests are usually *invited* by hosts and they are not strangers. An interesting observation is that guests may have different access policies or restrictions in their own local domains, e.g. *not* allowed to play a video clip on the TV in her/his own home, or a university does not want its researchers to connect to the other universities' wireless network. However, as guests, they are permitted to access those resources from other users who have such authorisation. Note that this is particularly significant. I will explain more about this later on.

- *Guests are not local users.* They do not have access to everything in foreign domains. An external consultant is not an employee from the client's organisation. Hence, the requests for accessing Smart Space services would be refused because the consultant's role in this client's organisation is not recognisable. We may give guests the right to access a DVD player in the living room but not for the PC in the studying area.

- *Guests are not pre-authorised users.* The requirements from guests are highly flexible. Consequently, it is infeasible to pre-define a general or individual guest user account, a role or a capability for them beforehand. In addition, the guests' permission may be changed dramatically under the hosts' willingness. We may allow adult guests to access DVD player but not for some under-14 years old guests. Even for the same guest, the host may change his/her permission on the access of some resources depending upon the nature of different contexts.

Thus, the scenario of *guest access* is defined as follows,

Guest Access: the access on some resources is said to be *guest access* only if the request is under the permission from authorised users, i.e. resource owners or other users who have corresponding access rights. Compared with *guests*, I will use a more general term, *authorised users* in the rest of this chapter. They are the users who have been authorised to access some resources either in their own domains or in other foreign domains.

6.5.2 Common Approaches for Guests Access

An essential concern for *guest access* is about how to give the appropriate permissions to guests. For authorised users, the "easiest" (unfortunately also the naive) way is to hand over to the guests their own capabilities. Semantically, it is similar to giving guests the door key for the house. It has major difficulty to revoke. It is important that the system can prevent guests from making their own copy of capabilities. Otherwise, the expensive process of changing the door lock has to be fulfilled every time after each guest's access. Also, it is cumbersome if the host wants to revoke the guest's access right immediately after the hand-over process completes.

Allowing the central system to set up a temporary credential in some form is another commonly used approach for *guest access*, e.g. a system account with a user name/password associating with the appropriate access rights. The

temporary credential will be de-activated as a result of the guest completing the necessary access. However, this approach does not address the essential problem for guest access in pervasive environments. In pervasive environments, guests are most likely from stranger domains (or foreign domains), and they may need different access rights to access some resources. This fact makes it more difficult for authorised users to assign different guests with appropriate access rights. This may end up with the problem that a guest may obtain a totally inappropriate right.

6.5.3 Motivating Threat Model - Guest Printing

Consider the following example. Darren is a professor from the University of Dolls. He is *invited* by Ellis, a senior lecturer in Computer Science School at East University, to give a seminar. Darren realises that he needs to print the presentation slides from his "DRD_D" when he notices a printer on the way to the seminar theatre.

The printer, P, cannot understand Darren's role in computer science, and therefore it denies his access. I assume that the printer P accepts the printing job from Ellis's "DRD_E" [5] according to the security policy in the school (e.g. 'any senior staff can access printers in the school'). Thus, the basic requirements of the security in this example are,

- The server in the Computer Science school (the owner of the printer P) will allow the guest, Darren, to access P, under the temporary permission from the authorised user, Ellis.

- For Darren, the right of printing is from Ellis and for the period that Ellis specifies.

- For the server in the Computer Science school, Darren's access is granted if,

[5] The DRD_E may be a stranger, or may not, from the printer viewpoint.

– Ellis has the appropriate access right to print;

– Ellis has the right to delegate her right (i.e. printing);

– Ellis's printing right is not revoked.

6.5.4 Introducing Guests

Instinctively, the guest printing is another example of access control over multiple domains. Hence, as learned from the domain-oriented approach, it naively appears Darren's local domain, the university of Dolls, ought to be involved to make the final decision. However, guests are not strangers. Instead, they are very often *invited* by some authorised users. In other words, guests are usually *friendly* players although they come from strange domains. The major task here is to make sure that Darren has the appropriate rights to do the printing job only.

Considering the specific requirements for granting the guest printing access listed above, the guest's (i.e. Darren) own local domain university Dolls is actually not responsible for his/her access. Ellis is responsible to *introduce* the guest to the resources, i.e. the printer P. Hence, Ellis makes the final decision in the guest printing example. In fact, it does not matter that Psychology school at the University of Dolls is saying "no" to the server of the computer science school at East University, perhaps because Darren just retired. The printer does not care as long as Ellis is happy to give Darren the permission to access the printer temporarily at this moment. Hence, knowing Darren's domain information does not help the printer to grant the guest access request. In other words, the printer does not concern itself with the domain Darren is from at all, "I do not care where the guest is from as long as he/she is introduced by Ellis.". Since the printer is managed by the server of the computer science school as well, thus, essentially, this guest printing scenario is more a case of access control for a single domain [6]. Thus, I argue that conventional

[6]The case where Ellis is a stranger from a friendly domain ca also be dealt with, by

access control methods can be used to address most guest access scenarios in pervasive environments. These methods have been described in section 2.2, and I do not intend to discuss the implementation in great detail here because this research mainly focuses upon the multiple domain context.

As a visiting professor, the guest Darren is *explicitly* trusted by Ellis. Consequently, Ellis is willing to let Darren access the printer P on a temporary basis as long as she knows what he is doing. This may be controversial in conventional environments. As described in Chapter 4, however, a pervasive environment is human-centred and more importantly the *human* context is the most significant consideration for pervasive environments. As human users, we have clear intention about what we are trying to accomplish or to avoid. A pervasive system should track human intent and the correct choice ought to depend upon this human context [108].

According to the school's security policy, Ellis can delegate access rights ("permission to print on the printer P") to anyone she trusts. Thus, she delegates to the guest Darren the right to use the printer P. Ellis's personal device DRD_E can send a signed delegation (e.g. the delegation certificates [10]) to Darren's PDA DRD_D. The two channel authentication mechanism can be implemented here to guarantee the signed delegation goes to Darren's device rather than that of anyone else in the proximity. Then, DRD_D sends his access request and the delegation to the printer P. The service owner, e.g. the server of the Computer Science school, will check the school's security policy, i.e. if Ellis's access right on P is still validated, and if Ellis is allowed to delegate this rights to the guests. If the delegation conforms to the policies, the server will send the "request to verify" (RTV) message to Ellis's DRD_E instead of granting Darren's access immediately,

"Are you aware that you are delegating *permission to print on the printer P* to a guest who is next to the printer P?"

combining the protocols in section 6.4 and section 6.5.

Note that the notification should include the *name* given to the guest *by Ellis* at the time of delegation.

The printer P allows Darren to print only after Ellis responds with the positive reply. Darren's access is on a temporary basis because this re-delegation is managed by Ellis. Hence, Ellis can revoke this re-delegation anytime by replying a negative message to the RTV message. Meanwhile, Darren cannot access services other than the printer P. When he needs to access other services, e.g. the digital projector at the seminar room, Darren must ask Ellis for another delegation. This scenario allows Darren, a guest, to access certain services in other domains without creating a pre-defined account, role, or identity for him.

6.6 Conclusions

This chapter provides a critical discussion for access control in LoT. It reflects the domain-based access control method proposed previously, and examines some issues relating to access control in pervasive environment in depth (both from multiple domain oriented and single domain oriented perspective). The crucial point is that this chapter gives the *details* of mechanisms to meet some of the requirements identified in earlier chapters.

The next chapter concludes this dissertation by providing a summary of contributions and some directions for future research.

Chapter 7

Conclusions and Future Work

Future pervasive computing applications will be of vast scale, and often intended to deal with complexly collaborative interactions from many human users or different organisations. A powerful design technique is to examine them from the *domain* perspective. Thus, *trust* can be reduced to a *local* level. This dissertation reviewed some previous work in a number of related areas, and examined and identified research issues that were yet to be addressed. A novel security framework for pervasive environments, LoT, has been developed. This chapter concludes this dissertation. It begins with the highlights of the main contributions of this work. Then, some future research directions are suggested. Finally, it provides a closing remark of this dissertation.

7.1 Summary of Contributions

The main contribution of this thesis is the proposal of a localisable, fully decentralised security framework – LoT. It is better suited to an open and heterogeneous environment, such as a multiple-domain-oriented pervasive environment, than a centralised computing environment. As a recapitulation of section 1.5, the following contributions have been made,

- Proposing the *localising the trust* security paradigm, serving as a simple guideline to design future security systems for multiple domains.

- Proposing the LoT framework as a generic model, thus providing a basic platform on which a tool-kit like set of security mechanisms from both *authentication* and *access control* ends can be built.

- Developing a 2-channel authentication protocol for pervasive environments instead of conventional (strong) ID-based authentication.

- Designing *profile certificate* as a important tool, realising the domain/local user related information conveyance.

- Designing an architecture for a domain-oriented Encryption-based Access Control approach, which enables flexible delegation and effective revocation in the multiple domain context.

- Designing an extended capability-based access control model, DuCaM, supporting the Encryption-based Access Control method.

7.2 Directions for Future Work

This thesis has developed a useful and "simple" framework for enabling spontaneous interactions, cross-domain authorisation and localised decision-making in the multiple domain context, i.e. pervasive environments. However, *multiple domain* for pervasive computing is a relatively new concept, and thus, this LoT framework presents an excellent basis for further exploration in this area, particularly for providing security services in many pervasive computing applications.

The detailed approaches and protocols demonstrated in this dissertation are techniques which can be generalised. LoT only gives the example of two layers, i.e. domains (domain server) and local users. This technique can be generalised recursively, or in a more hierarchical way, with respect to different applications. Moreover, it can also be more flexible by allowing different

domains having different layers. This will be decided by specific security requirements for particular applications. A player Alice may be required to get the access permission not only from her own department, but also from the financial department for example. Thus, the LoT framework will be adjusted to require two pieces of partial decryption key. One of them is supplied by Alice's department server, and the other need to be provided by the server for the financial department.

Some selected future work is suggested in this section. As pointed out in the beginning of chapter 2 (see page 24), we cannot just simply copy the LoT framework to address *problems* in different contexts. As a result, we discuss not only a number of potential extensions built from the platform of LoT, but most importantly some areas that would potentially benefit (to solve the problems that they target) from generalising the techniques that have been deployed in the development of this research. In addition, we shall be pointing out some work that we would *not* want to be seen done.

7.2.1 Authentication and Expiration

The need of terminating authentication has been raised by Schneier in [110] (see a brief discussion in section 2.1.2, page 27). On-line shopping is a typical example. On one hand, users worry about the leakage of their personal details (i.e. address, credit card details) which are stored in a database somewhere. On the other hand, users will have to make relevant change in every single website that they have an account with, when their details change (e.g. moving house, expired credit card).

One way addressing this problem is to allow users to manage their own information, following the "localising the trust" security paradigm. Briefly speaking, a trusted *user domain server* is introduced to the loop. Based upon users' self-defined policy, it can be a desktop PC, home server (used to control all the pervasive devices at home), or a trusted website. A user Alice put

her (encrypted) details in this domain server. When she intends to purchase some books on-line using her PDA or public computer in a local cafe, she only sends a request to amazon.com. This request should contain information which is enough for the server from amazon.com to know where (i.e. which user domain server) to contact. Essentially, Alice is authenticated by her own domain. The transaction is completed when amazon.com receives a positive reply from Alice's domain server [1].

Thus, Alice only needs to change details, such as the expiry date of her new credit card, stored in her own domain server. More importantly, if she has a transaction with a malicious website, this website will not retrieve her secret personal details in any way.

7.2.2 Capability-based Access Control

Capability-based access control has been one of our major interests throughout this research. Surprisingly, the research on capability-based access control has earned little attention after the many capability-based access control systems developed before the middle 90s. Arguably, it is probably because capability-based access control has been fully-investigated and well-developed for a single domain. Now, the challenging multiple domain applications have opened a new opportunity for many early capability-based systems, for instance, Tanenbaum and Mullender's Amoeba [89]. For the original F-boxes in Amoeba, they shared the same information. Hence, once an attacker breaks into one F-box, the security for the entire Amoeba is broken. Amoeba is not alone on this matter, unfortunately, this is a typical security problem for other early capability-based systems (even for Gong's multi-domain I-CAP [57]).

In this dissertation (see section 5 and 6), we showed the techniques to adapt I-CAP [57] using the domain-oriented encryption-based access control

[1] For the purpose of simplicity, the bank's server is not considered here. In practice, however, the involvement of a bank's server will not necessarily complicate the protocol run.

method. We also use the *domain-level* ACL and *localised* authentication, which shares with S-CAP [69] a common understanding that being in an access control list is necessary but not sufficient to gain access, and users have to authenticate themselves again. Those techniques used in the LoT framework can also be applied to Amoeba. We may think of placing those (tamper resistant and trusted) F-boxes in each domain. Capabilities that are output from an F-box should provide sufficient (but just enough) domain information (i.e. which domain or F-box the capability holder is from) to another F-box. User authentication, again, ought to be localised within a domain. A potential benefit of doing so is the damage control. If one domain is broken, for example because its F-boxes are compromised, it is not the end of the world.

When a player intends to access some resources in a remote domain, he submits his capability to the remote domain's domain server. The remote domain server generates a token by inputting the player's capability to the remote domain's F-box. Then, the remote F-box gives the player this token. If the player is from the correct domain, the player should be able to convert this token to the useful access information by satisfying his local domain's authentication mechanism. Specifically, a piece of information needs to be retrieved from the player's local domain server. Moreover, if a domain would like to revoke its local players' capabilities, the only thing the domain server needs to do is to stop issuing this piece of information to its local players at the time when transactions occur. Consequently, breaking into one F-box in one domain does not help an attacker to subsequently break the entire system. Each capability is constrained within one domain, and ease of revocation is achieved by implementing domain-based encryption techniques.

Similarly, those techniques can also be deployed in classical Kerberos [86] system. Differing from conventional Kerberos tickets, a new type of ticket is defined to indicate which Kerberos server the ticket holder (let us say Alice)

is from. Resource servers will hand over Alice a token based on domain information in Alice's ticket. If Alice is from correct domain and delegated with correct access right, the token can be converted to a useful access capability with her domain Kerberos server's assistance. Thus, Alice is thought to be a correct stranger if the conversion is completed.

7.2.3 Context-Awareness Applications in Pervasive Computing

Some existing security systems have made use of the context-awareness side of pervasive computing. Kagal's trust-based system [67] is one of them, which addresses security issues for pervasive computing by adding location-aware technology. But it does not solve the inherent certificate revocation problem. LoT could be used to address this problem by separating (conventional) certificates from final access decision-making. Ultimately, the access decision would be made by local domains, following the *localising trust* design principle.

Moreover, it is noted that context is not just limited to physical location. In fact, the context-awareness service is the awareness of the physical environment, such as, location, audio, images or any other physical conditions in surrounding environments. The authentication protocols described in chapter 4 represent an attempt at applying different contexts (rather than just location) to deal with situations where stranger devices are involved. This attempt may be worthwhile to be extended to other security systems which use context-awareness as a key security parameter.

LoT does not currently address the *privacy* issue. However, privacy is an important factor for many context-awareness applications. This may be an area that can be explored further. Again, localising trust can play an important role in the privacy-oriented pervasive applications. A player's personal information is only visible and available to his own domain. Foreign domains/players do not need to know this information, because the final

decision-making is controlled by the player's local domain. They even do not care about those personal information, "who you are", "where you are", as long as domains are happy with their local players' request.

7.2.4 Work that We Do Not Want To See

LoT introduces different tools to provide different security services, particularly for authentication and authorisation, unlike the swiss army knife. The methods and approaches in this tool-kit should not be treated separately. However, I do not want to see the integration of those tools into a single tool. As witnessed from many existing security systems, a security protocol designed for providing one security service may be less than helpful when deployed for a totally different security purpose.

Thus, having a tool-kit in a security framework will avoid such a misuse by enforcing security countermeasures to answer correct questions. More importantly, this approach will include human users into the loop, particularly in pervasive environments as described in chapter 4. The positive human context ("we know what we are doing") will significantly leverage the security in pervasive environments

7.3 Conclusions

This dissertation has presented LoT, a fully localised security framework designed for fast growing pervasive environment applications. It is independent of any global trust infrastructure. Arguably, some (heavy)setup work is involved at the beginning, (this is the cost of doing business) i.e. delegation between domain servers and local users. However, once this is configured, LoT is extremely efficient and effective to address some security issues in pervasive environments.

The crucial novelty of LoT lies in its **architectural innovation**. It

changes the way of thinking about security by introducing *multiple domains* as a primary concern, but leaving the basic underlying security protocols (for a single domain) and system implementation untouched. Although much future research remains to be done, as studied and demonstrated in this dissertation, the techniques developed by LoT is a fruitful way forward in future research for security in pervasive environments.

Bibliography

[1] M. Abadi, M. Burrows, B. Lampson, and G. Plotkin. A Calculus for Access Control in Distributed Systems. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 15(4):706 – 734, September 1993.

[2] S.S. Al-Riyami and K.G. Paterson. Certificateless Public Key Cryptography. In C.S.Laih, editor, *Proc. ASIACRYPT 2003*, LNCS 2894, pages 452 – 473. Springer, 2003.

[3] R. Anderson. Why Cryptosystems Fail. *Communications of the ACM*, 37(11):32 – 40, November 1994.

[4] R. Anderson. *Security Engineering*. John Wiley & Sons, Inc., 2001.

[5] R. Anderson, F. Bergadano, B. Crispo, J. Lee, C. Manifavas, and R. Needham. A New Family of Authentication Protocols. *ACM SIGOPS Operating Systems Review*, 32(4):9 – 20, October 1998.

[6] R. Anderson, M. Bond, and S. J. Murdoch. Chip and Spin. http://www.chipandspin.co.uk/, 2005.

[7] R. Anderson, C. Manifavas, and C. Sutherland. Netcard – A Practical Electronic Cash System. In *4th Cambridge Security Protocols Workshop*, 1996.

[8] J. Arkko and P. Nikander. Weak Authentication: How to Authenticate Unknown Principals without Trusted Parties. In *Security Protocols: 10th International Workshop, Cambridge, UK*, LNCS2845, pages 5 – 19. Springer - Verlag Berlin Heidelberg, 2004.

[9] N. Asokan and P. Ginzboorg. Key Agreement in Ad-hoc Networks. *Computer Communication Review*, (23):1627 – 1637, 2000.

[10] T. Aura. Distributed Access-Rights Managements with Delegations Certificates. In *Secure Internet Programming*, pages 211–235, 1999.

[11] J. Bacon, K. Moody, and W. Yao. A Model of OASIS Role-Based Access Control and Its Support for Active Security. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):492–540, November 2002.

[12] D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and Hao-Chi Wong. Secret Handshakes from Pairing-Based Key Agreements. In *24th IEEE Symposium on Security and Privacy*, Oakland, CA, May 2003.

[13] D. Balfanz, D. Smetters, P. Stewart, and H. Wong. Talking to Strangers: Authentication in ad-hoc Wireless Networks. In *Symposium on Nework and Distributed Systems Security (NDSS'02)*, February 2002.

[14] S. Berkovits, S. Chokhani, J. A. Furlong, J. A. Geiter, and J. C. Guild. Public Key Infrastructure Study. Mitre technical report, National Institute of Standards and Technology, April 1994.

[15] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The Keynote Trust Management System Version. Internet RFC 2704, September 1999.

[16] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized Trust Management. In *Proc. IEEE Conference on Security and Privacy*, pages 164–173, Oakland, CA, May 1996.

[17] M. Bond and G. Danezis. The Dining Freemasons (Security Protocols for Secret Societies). In *13th International Workshop on Security Protocols*, April 2005. To appear.

[18] D. Boneh and M. Franklin. Identity based Encryption from the Weil Pairing. In *Advances in Cryptology - ASIACRYPT 2001*, pages 213–229. Springer-Verlag, 2001.

[19] M. Burrows, M. Abadi, and R. Needham. A Logic of Authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.

[20] V. Cahill, E. Gray, J.-M. Seigneur, C.D. Jensen, Y. Chen, B. Shand, N. Dimmock, A. Twigg, J. Bacon, C. English, W. Wagealla, S. Terzis, P. Nixon, G. Di Marzo Serugendo, C. Bryce, M. Carbone, K. Krukow, and M. Nielsen. Using Trust for Secure Collaboration in Uncertain Environments. *Pervasive Computing*, 2(3):52–61, September 2003.

[21] G. Candea and A. Fox. Using Dynamic Mediation to Integrate Cots Entities in a Ubiquitous Computing Environment. In *Second International Symposium on Handheld and Ubiquitous Computing*, pages 248 – 254, 2000.

[22] S. Capkun, J. Hubaux, and L. Buttyan. Mobility Helps Security in Ad Hoc Networks. In *MobiHoc'03*, June 2003.

[23] D. Chaum, A. Fiat, and M. Naor. Untraceable Electronic Cash. In *Advances in Cryptology - Crypto '88*, pages 319 – 327. Springer, 1990.

[24] L. Chen, K. Harrison, A. Moss, D. Soldera, and N. P. Smart. Certification of Publik Keys within an Identity Based System. In A. H. Chan and V. Gligor, editors, *ISC 2002*, volume 2433/2002, pages 322–333. Springer-Verlag Heidelberg, 2002.

[25] B. Christianson and W. S. Harbison. Why Isn't Trust Transitive? In *Proceedings of the International Workshop on Security Protocols*, LNCS 1189, pages 171–176. Springer-Verlag, 1997.

[26] B. Christianson and J. Li. Multi-channel Key Agreement using Encrypted Public Key Exchange. In *15th Security Protocols Workshop*, April 2007. To appear.

[27] B. Christianson and J. A. Malcolm. Binding Bit Patterns to Real World Entities. In *Proceedings of the 5th International Workshop on Security Protocols*, LNCS 1361, pages 105–113. Springer-Verlag, 1998.

[28] B. Christianson, M. Roe, and D. Wheeler. Secure Sessions from Weak Secrets. In *Proceedings of the 11th International Workshop on Security Protocols*, LNCS 3364. Springer-Verlag, 2003.

[29] D. Clarke, B. Gassend, T. Kotwal, M. Burnside, M. van Dijk, S. Devadas, and R. Rivest. The Untrusted Computer Problem and Camera-Based Authentication. In F.Mattern and M.Naghshineh, editor, *Pervasive 2002*, pages 114 – 124. Springer - Verlag Berlin Heidelberg, 2002.

[30] J. Cluclow and T. Moore. Suicide for the Common Good: a New Strategy for Credential Revocation in Self-Organizing Systems. *ACM SIGOPS Operating Systems Reviews*, 40(3):18–21, July 2006.

[31] M. D. Corner and B. D. Noble. Zero-Interaction Authentication. In *The 8th ACM Conference on Mobile Computing and Networking (MobiCom'02)*, pages 1–11, September 2002.

[32] M. J. Covington, M. R. Sastry, and D. J. Manohar. Attribute-Based Authentication Model for Dynamic Mobile Environments. In *Security of Pervasive Computing (SPC 2006)*, LNCS 3934, pages 227–242, 2006.

[33] S. Creese, M. Goldsmith, B. Roscoe, and M. Xiao. Bootstrapping Multi-Party Ad-Hoc Security. In *Proceedings of the 2006 ACM symposium on applied computing (SAC'06)*, pages 369 – 375. ACM Press, 2006.

[34] S. Creese, M. Goldsmith, B. Roscoe, and I. Zakiuddin. The Attacker in Ubiquitous Computing Environments: Formalising the Threat Model. In *In Proc. of the 1st International Workshop on Formal Aspects in Security and Trust*, pages 83 – 97, 2003.

[35] S. Creese, M. Goldsmith, B. Roscoe, and I. Zakiuddin. Authentication for Pervasive Computing. In D. Hutter et al., editor, *Security in Pervasive Computing 2003*, LNCS 2802, pages 116 – 129. Springer-Verlag Berlin Heidelberg, 2004.

[36] B. Crispo. *Delegation of Responsibility*. Ph.D thesis, Wolfson College, the University of Cambridge, May 1999.

[37] B. Crispo and T. M. A. Lomas. A Certification Scheme for Electronic Commerce. In *Proceedings of the 3rd International Workshop on Security Protocols*, pages 19 – 32, 1996.

[38] P. Das Chowdhury. *Anonymity and Trust in Electronic World*. Ph.D thesis, University of Hertfordshire, February 2005.

[39] P. Das Chowdhury, B. Christianson, and J. Malcolm. Anonymous Context Based Role Activation Mechanism. In *Proceedings of the 13th International Workshop on Security Protocols*. Springer-Verlag, April 2005.

[40] P. J. Denning. Fault Tolerant Operating Systems. *ACM Computing Surveys (CSUR)*, 8(4):359–389, December 1976.

[41] W. Diffie and M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[42] T. ElGamal. A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in Cryptology: Proceedings of CRYPTO 84'*, pages 10–18. Springer-Verlag, 1985.

[43] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. M. Thomas, and T. Ylonen. SPKI Certificate Theory. Internet RFC 2693, October 1999.

[44] L. Eschenauer, V. D. Gligor, and J. Baras. On Trust Establishment in Mobile Ad-Hoc Networks. In *Security Protocols Workshop 2002*, volume LNCS 2845, pages 47–66. Springer-Verlag, 2004.

[45] L. M. Feeney, B. Ahlgren, and A. Westerlund. Spontaneous Networking: An Application-Oriented Approach to Ad Hoc Networking. *IEEE Communications Magazine*, pages 176–181, June 2001.

[46] A. Fox and S. D. Gribble. Security on the Move: Indirect Authentication using Kerberos. In *Procceedings of the 2nd Annual International Conference on Mobile Computing and Networking*, pages 155 – 164. ACM Press, 1996.

[47] B. Fox and B. LaMacchia. Certificate Revocation: Mechanics and Meaning. In *Financial Cryptography 98*, LNCS 1465, pages 158–164. Springer Berlin / Heidelberg, 1998.

[48] S. Galice, V. Legrand, M. Minier, J. Mullins, and S. Ubeda. The KAA Project: a Trust Policy Point of View, 2006.

[49] M. Gasser and E. McDermott. An Architecture for Practical Delegation in a Distributed System. In *IEEE Symposium on Security and Privacy*, pages 20 – 30, 1990.

[50] C. Gehrmann, C.J. Mitchell, and K. Nyberg. Manual Authentication for Wireless Devices. *Cryptobytes*, 7(1):29 – 37, 2004.

[51] C. Gentry. Certificate-based Encryption and the Certificate Revocation Problem. In E. Biham, editor, *Proc. EUROCRYPT 2003*, LNCS 2656, pages 272–293. Springer, 2003.

[52] C. Gentry and A. Silverberg. Hierarchical ID-based Cryptography. In Y. Zheng, editor, *Proc. ASIACRYPT 2002*, LNCS 2501, pages 548–566. Springer, 2002.

[53] L. Giuri. Role-based Access Control: A Natural Approach. In *Proceedings of the first ACM Workshop on Role-based access control*, pages 33–37, 1996.

[54] O. Goldreich, B. Pfitzmann, and R. L. Rivest. Self-delegation with Controlled Propagation — or — what if you lose your laptop. *Lecture Notes in Computer Science*, 1462:153–168, 1998.

[55] L. Gong. A Secure Identity-based Capability System. *IEEE symposium on security and privacy*, pages 56–65, 1989.

[56] L. Gong. Using One-Way Functions for Authentication. *ACM Computer Communication Review*, 19(5):8–11, October 1989.

[57] L. Gong. *Cryptographic Protocols for Distributed Systems*. Ph.D thesis, University of Cambridge, 1990.

[58] L Gong, T. M. A. Lomas, R. M. Needham, and J. H. Saltzer. Protecting Poorly Chosen Secrets From Guessing Attacks. *IEEE Journal on Selected Areas in Communications*, 11(5):648–656, June 1993.

[59] C. A. Gunter and T. Jim. Generalized Certificate Revocation. In *POPL '00: Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 316–329, 2000.

[60] S. Halevi, P. A. Karger, and D. Naor. Enforcing Confinement in Distributed Storage and a Cryptographic Model for Access Control. Eprint archive, eprint.iacr.org/2005/169.

[61] U. Hengartner and P. Steenkiste. Exploiting Hierarchical Identity-Based Encryption for Access Control to Pervasive Computing Information. In *Proc. of SecureComm 2005*, pages 384–393, September 2005.

[62] A. Herzberg, Y. Mass, J. Mihaeli, D. Naor, and Y. Ravid. Access Control Meets Public Key Infrastructure, Or: Assigning Roles to Strangers. In *IEEE Symposium on Security and Privacy 2000 (S&P 2000)*, pages 2 – 14. IEEE Computer Society, 2000.

[63] J. Holt, R. Bradshaw, K. Seamons, and H. Orman. Hidden Credentials. In *2nd ACM Workshop on Privacy in the Electronic Society*, pages 1 – 8. ACM Press, October 2003.

[64] J. Horwitz and B. Lynn. Toward Hierarchical Identity-Based Encryption. In L. R. Knudsen, editor, *EUROCRYPT 2002*, pages 466 – 481. Springer - Verlag Berlin Heidelberg, 2002.

[65] R. Housley, W. Ford, W. Polk, and D. Solo. RFC2459: Internet X.509 Public Key Infrastructure Certificate and CRL Profile. Internet RFCs, 1999.

[66] M. Johnson and F. Stajano. Usability of Security Management: Defining the Permission of Guests. In *Security Protocols Workshop 2006*, 2006. To appear.

[67] L. Kagal, T. Finin, and A. Joshi. Trust-based Security in Pervasive Computing Environment. *Computer*, 34(12):154 – 157, December 2001.

[68] L. Kagal, J. Undercoffer, F. Perich, A. Joshi, T. Finin, and Y. Yesh. Vigil: Providing Trust for Enhanced Security in Pervasive Systems. Technical report, University of Maryland Baltimore County, August 2002.

[69] P. A. Karger. *Improving Security and Performance for Capability Systems*. Ph.D thesis, University of Cambridge, 1988.

[70] P. A. Karger. New Methods for Immediate Revocation. In *IEEE Symposium on Security and Privacy*, pages 48–55, 1989.

[71] P. A. Karger and A. J. Herbert. An Augmented Capability Architecture to Support Lattice Security and Traceability of Access. In *the IEEE Symposium on Security and Privacy*, pages 2 – 12, 1984.

[72] A. H. Karp, G. J. Rozas, A. Banerji, and R. Gupta. Using Split Capabilities for Access Control. *IEEE Software*, 20(1):42–49, January 2003.

[73] H. Khurana and V. D. Gligor. Review and Revocation of Access Privileges Distributed with PKI Certificates. In *the 8th International Security Protocols Workshop*, volume LNCS 2133, pages 100–112. Springer, April 2000.

[74] B. Lampson, M. Abadi, M. Burrows, and E. Wobber. Authentication in Distributed Systems: Theory and Practice. *ACM Trans. Computer Systems*, 10(4):265 – 310, November 1992.

[75] B. W. Lampson. Protection. *ACM SIGOPS Operating Systems Review*, 8(1):18–24, January 1974.

[76] B.W. Lampson. A Note on the Confinement Problem. *Communications of the ACM*, 16(10):613 – 615, 1973.

[77] M. Langheinrich. When Trust does not Compute – The Role of Trust in Ubiquitous Computing. Workshop on Privacy at Ubicomp 2003, October 2003.

[78] J. Li, B. Christianson, and M. Loomes. Fair Authentication in Pervasive Computing. In *Secure Mobile Ad-hoc Networks and Sensors (MADNES'05)*, volume LNCS 4074, pages 132 – 143. Springer Berlin/Heidelberg, August 2006.

[79] N. Li and J. Feigenbaum. Nonmonotonicity, User Interfaces, and Risk Assessment in Certificate Revocation (Position Paper). In *Proceedings of the Fifth International Conference on Financial Cryptography (FC'01)*, volume LNCS 2339, pages 166–177. Springer, February 2001.

[80] M. R. Low. *Self Defence in Open Systems: Protecting and Sharing Resources in a Distributed Open Environment*. Ph.D thesis, University of Hertfordshire, September 1994.

[81] M.R. Low and B. Christianson. Self Authenticating Proxies. *Computer Journal*, 37(5):422–428, October 1994.

[82] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, Florida, 1997.

[83] S. Micali. Efficient Certificate Revocation. Technical Report TM-542b, M.I.T., Cambridge, MA, USA, 1996.

[84] S. Micali. NOVOMODO: Scalable Certificate Validation and Simplified PKI Management. In *1st Annual PKI Research Workshop*, pages 15–25, April 2002.

[85] S. Milgram. The Small World Problem. *Psychology Today*, 1:61, 1967.

[86] S. P. Miller, B. C. Neuman, J. I. Schiller, and J. H. Saltzer. Kerberos Authentication and Authorisation System. Project Athena Technical Plan, section e.2.1, M.I.T, October 1988.

[87] K. Mitnick. *The Art of Deception: Controlling the Human Element of Security*. Wiley, October 2002.

[88] T. Moore, J. Cluclow, R. Anderson, and S. Nagaraja. New Strategies for Revocation in Ad-Hoc Networks. In *the Fourth European Workshop on Security and Privacy in Ad Hoc and Sensor Networks (ESAS)*, volume LNCS 4572, pages 232–246. Springer, July 2007.

[89] S. J. Mullender. *Principles of Distributed Operating System Design*. Ph.D thesis, Vrije Universiteit te Amsterdam, 1985.

[90] M. Myers. Revocation: Options and Challenges. In *FC '98: Proceedings of the Second International Conference on Financial Cryptography*, pages 165–171. Springer-Verlag, 1998.

[91] M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. IETF RFC 2560, June 1999.

[92] R. Needham. Keynote Address: Security Protocols and the Swiss Army Knife. In B.Christianson, B.Crispo, and J.A.Malcolm et al., editors, *Security Protocols: 8th International Workshops*, volume 2133, pages 1–4. Springer-Verlag, January 2001.

[93] R. M. Needham and M. D. Schroeder. Using Encryption for Authentication in large Networks of Computers. *Communications of the ACM*, (21(12)):993–999, 1978.

[94] B.C. Neuman. Proxy-Based Authorisation and Accounting for Distributed Systems. In *the 13th International Conference on Distributed Computing Systems*, pages 283–291, May 1993.

[95] S. Osborn, R. Sandhu, and Q. Munawer. Configuring Role-based Access Control to Enforce Mandatory and Discretionary Access Control Policies. *ACM Transactions on Information and System Security*, 3(2):85 – 106, 2000.

[96] S. Pancho. Paradigm Shifts in Protocol Analysis. In *Proceedings of the 1999 Workshop on New Security Paradigms*, pages 70–79, 1999.

[97] J. Riordan and B. Schneier. Environmental Key Generation Towards Clueless Agents. In *Mobile Agents and Security*, volume LNCS 1419, pages 15 – 24. Springer-Verlag Berlin Heidelberg, 1998.

[98] R. L. Rivest. Can We Eliminate Certificate Revocation Lists? In *Proceedings of Financial Cryptography 98*, volume LNCS 1465, pages 178–183, February 1998.

[99] R. L. Rivest and B. Lampson. SDSI - A Simple Distributed Security Infrastructure. Presented at CRYPTO'96 Rumpsession, 1996.

[100] R. L. Rivest and A. Shamir. PayWord and MicroMint: Two Simple Micropayment Schemes. In *4th Cambridge Security Protocols Workshop 1996*, pages 69 – 87, 1996.

[101] R. L. Rivest, A. Shamir, and L. M. Adleman. A Method for Obtaining Digital Signatures and Public-key Cryptosystems. *Communications of the ACM*, v. 21(n. 2):120–126, February 1978.

[102] R. L. Rivest, A. Shamir, and Y. Tauman. How to Leak a Secret. In *Advances in Cryptology - ASIACRYPT 2001*, volume LNCS 2248, pages 552–565, December 2001.

[103] M. Roe. *Cryptography and Evidence*. Ph.D thesis, University of Cambridge, 1997.

[104] D. Saha and A. Mukherjee. Pervasive Computing: A Paradigm For The 21st Century. *IEEE Computer*, 36(3):25 – 31, March 2003.

[105] J. H. Saltzer and M. D. Schroeder. The Protection of Information in Computer Systems. In *Proceedings of the IEEE*, volume 63, pages 1278–1308, September 1975.

[106] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based Access Control Models. *Computer*, 29(2):38 – 47, February 1996.

[107] R.S. Sandhu and P. Samarati. Access Control: Principles and Practice. *IEEE Communications Magazine*, 32(9):40 – 48, 1994.

[108] M. Satyanarayanan. Pervasive Computing: Vision and Challenges. {*IEEE*} *Personal Communications*, (10-17), August 2001.

[109] B. Schneier. *Secrets & Lies*. Wiley Computer Publishing, 2000.

[110] B. Schneier. Authentication and Expiration. *IEEE Security and Privacy*, 3(1):88, 2005.

[111] C. P. Schnorr. Efficient Signature Generation by Smart Cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[112] J. M. Seigneur, S. Farrell, C. D. Jensen, E. Gray, and Y. Chen. End-to-End Trust Starts with Recognition. In D. Hutter et al., editor, *Security in Pervasive Computing 2003*, LNCS 2802, pages 130 – 142. Springer - Verlag Berlin Heidelbery, 2004.

[113] A. Shamir. Identity-based Cryptosystems and Signature Schemes. In *Lecture Notes in Computer Science*, volume 196, pages 47–53. in Advances in Cryptology - CRYPTO '84, Springer-Verlag, 1984.

[114] B. Shand, N. Dimmock, and J. Bacon. Trust for Ubiquitous, Transparent Collaboration. *Wireless Networks*, 10(6):711–721, 2004.

[115] J. S. Shapiro. *EROS: A Capability System*. Ph.D thesis, University of Pennsylvania, 1999.

[116] N. P. Smart. An Identity Based Authenticated Key Agreement Protocol Based on the Weil Pairing. *Electronic Letters*, (38(13)):630–632, 2002.

[117] N.P. Smart. Access Control Using Pairing Based Cryptography. In M. Joye, editor, *Proceedings CT-RSA 2003*, volume 2612 of *LNCS*, pages 111–121. Springer, 2003.

[118] K. R. Sollins. Cascaded Authentication. In *Procceeding of the 1988 IEEE Symposium on Security and Privacy*, pages 156–163, 1988.

[119] F. Stajano. Security for Whom? The Shifting Security Assumptions of Pervasive Computing. In *Proceedings of International Security Symposium*, volume 2609. Springer-Verlag, 2002.

[120] F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In B. Christianson, B. Crispo, and M. Roe, editors, *Security Protocols, 7th International Workshop Proceedings, Lecture Notes in Computer Science*, LNCS 1296, pages 172 – 194, 1999.

[121] F. Stajano and R. Anderson. The Resurrecting Duckling: security issues for ubiquitous computing. *IEEE Computer*, 35(4):22–26, April 2002.

[122] J.G. Steiner, C. Neuman, and J.I. Schiller. Kerberos: An Authentication Service for Open Network Systems. In *Proceedings of the Winter 1988 Usenix Conference*, February 1988.

[123] S.G. Stubblebine. Recent-secure authentication: enforcing revocation in distributed systems. In *IEEE Symposium on Security and Privacy*, pages 224–235, 1995.

[124] Y. S. Tsiounis. *Efficient Electronic Cash: New Notions and Techniques*. Ph.d thesis, Northeastern University, June 1997.

[125] A.M. Turing. Computing Machinery and Intelligence. *MIND*, 49:433 – 460, 1950.

[126] L. von Ahn, M. Blum, N. J. Hopper, and J. Langford. CAPTCHA: Using Hard AI Problems for Security. In *Advances in Cryptology, Eurocrypt 2003*, volume 2656, pages 294 – 311, May 2003.

[127] L. von Ahn, M. Blum, and J. Langford. Telling Humans and Computers Apart Automatically. *Communications of the ACM*, 47(2):56 – 60, 2004.

[128] M. Weiser. The Computer for the Twenty-First Century. *Scientific American*, 265(3):94 – 104, September 1991.

[129] M. Weiser. Some Computer Science Issues in Ubiquitous Computing. *Communications of the ACM*, 36(7):75–84, July 1993.

[130] F.L. Wong and F. Stajano. Multi-Channel Protocols: Strong Authentication Using Camera-Equipped Wireless Devices. In *Proceedings of the13th International Workshop on Security Protocols*, April 2005.

[131] W. Yao. *Trust Management for Widely Distributed Systems*. Ph.D thesis, University of Cambridge, 2004.

[132] P. R. Zimmermann. *The Official PGP User's Guide*. MIT Press, Cambridge, MA, 1995.