A trust-based access control scheme for social networks

Wilfred Villegas

School of Computer Science
McGill University
Montreal, Canada

October 2008

A thesis submitted to McGill University in partial fulfillment of the requirements for the degree of Master of Science.

Dedication

For Mom, Dad, and Wibb.

Acknowledgments

I would like to thank my supervisor, Professor Muthucumaru Maheswaran, for his guidance and the many discussions which encouraged me to think critically and creatively. This thesis would not be possible without him.

I would also like to thank my colleagues at the Advanced Networking Laboratory (ANRL) who made my time with them enjoyable and rewarding. In particular, I would like to thank Bader Ali for several insightful discussions and suggestions, and for helping out with the simulations.

I would also like to thank Said Al-Shaqsi and Hui Guo for helping out with programming, and Steven Bidd and Louis Vigneault for translating the abstract into French.

I would like to thank my friends and especially Joyce for making my time spent in Montreal a truly memorable experience. Finally, I would like to thank my mother, Erlinda, my father Wilfredo, and my brother Wilbert. Their continuing love, support, and patience encouraged me to pursue my dreams.

Abstract

The personal data being published on online social networks is presenting new challenges in sharing of this digital content. This thesis proposes an access control scheme called *Personal Data Access Control*, or PDAC, which allows users to share data among their friends, using a trust computation to determine which friends should be given access. This trust computation uses previous interactions among a user's friends to classify his or her peers into one of three protection zones, which determine whether that peer gains access to the user's data. Additionally, the user may designate certain friends as attesters who will aid the user in determining which peers are trustworthy enough to be given access to his or her data. Simulations of the PDAC scheme were performed to evaluate its effectiveness in enforcing data access privileges. The results show that PDAC preserves confidentiality by exploiting the trust that is captured in existing social networks.

Résumé

Les données personnelles publiées sur internet par l'entremise des nouveaux réseaux sociaux virtuels présentent des défis considérables en ce qui attrait à l'échange numérique. Cette thèse propose un système de contrôle d'accès appelé *Personal Data Access Control*, ou PDAC, qui permet aux utilisateurs d'échanger leurs données personnelles avec leurs amis de façon mesurée, en utilisant un calcul de confiance. Ce calcul de confiance utilise comme critères d'évaluation les interactions antérieures entre l'utilisateur et chacun de ses amis afin de classer chacune de ses connaissances dans une de trois zones de protection. Ces zones délimitent le niveau d'accès accordé aux données de l'utilisateur. De plus, l'utilisateur peut assigner certains amis come vérificateurs qui donnent leur approbation et ainsi détermine en toute confidentialité qui devrait avoir accès a ses données. Nos résultats d'analyse démontrent que le PDAC accorde privilèges d'accès aux données de façon efficace. Ces simulations démontrent aussi que le PDAC préserve la confidentialité en saisissant les niveaux de confiance qui existe dans les réseaux sociaux virtuels d'aujourd'hui actuels.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

User-generated digital content on the Internet is proliferating at an unprecedented rate [1]. The immense volume of data that is being created is presenting new challenges in indexing, archiving, searching, browsing, and sharing of this digital content. This thesis is concerned with one of these key issues, namely the sharing of digital content.

1.1 Motivations

Online social networks have experienced a surge in popularity recently, particularly because they allow users to share music, photographs, home movies, and blogs with friends and family quickly and easily. Two of the most popular social networks today, `myspace.com` and `facebook.com`, allow users to easily share their data with their family, friends, coworkers, and classmates. The access control mechanism used by these networks to share data is based primarily on relationship depth (*friend*, *friend-of-a-friend*, etc.), organization membership (such as a workplace or school), or, as is the case with `facebook.com`, geographic location (such as a city or country network).

This protection scheme, while straightforward, assumes that all friends are equal. Clearly, this is not always the case in real life. These social networks fail to account for the fact that some friends may be more trusted than others. More fine-grained control is needed in the access control mechanisms currently employed by social networks.

A simple solution for this problem would be to use access control lists that explicitly denote which users are allowed access to the data [2]. While this approach gives the owner of the data the fine-grained access control we seek, it requires that the owner assign access rights to each friend that should see the data. This can be time consuming and repetitive for the owner, and the amount of effort required for data management can scale up as a product of the number of data objects and the owner's friend list. In order to deal with the rapidly increasing amount of user-generated data, we must develop protection techniques that minimize the amount of effort required to manage such large volumes of information.

Another major issue with respect to the publication of data on a social network is that of security among those that have gained access to a data object. Security settings usually apply to the data item only, and once a user has gained access to a copy, the original security settings no longer apply to the copy. Suppose that a user Bob has gained access to a data item that has been published by Alice on a social network. Assume that Bob satisfied the requirements of whatever system is controlling access to the data item. Current social network implementations do not prohibit Bob from reposting the data item under weaker access constraints, essentially rendering Alice's original constraints useless. Oscar, a malicious user who was not able to access the data item under Alice's original constraints, may be able to gain access through Bob's reposting of that item.

1.2 Addressing The Issues

This thesis proposes an access control scheme that seeks to address these issues. *Personal Data Access Control*, or PDAC, allows users to share data among their friends, using a trust computation to determine which friends should be given access. This trust computation will be based upon the recent interactions between any two users. Just as in real life relationships, the trust levels can vary from friend to friend, and may change over time. The changes in trust are stored in the social network. PDAC defines three distinct protection zones: the *Acceptance Zone*, the *Attestation Zone*, and the *Rejection Zone*. The zone boundaries are defined by the owner of the data to be protected. All users seeking to access the data are mapped into one of the three zones. Users which are mapped onto the Acceptance Zone get unconditional access to the protected data object. Conversely, requests from users which are mapped onto the Rejection Zone are denied access. Requests from users which are mapped onto the Attestation Zone are validated on a per-request basis. The complete PDAC scheme will be presented and we shall see that PDAC provides a flexible protection mechanism that is user friendly and requires low management effort.

Because the trust between two peers are captured by the social network, the scheme minimizes the need for access control lists, greatly reducing the amount of user intervention that is required. Nominal use of the access control scheme is independent of the number of friends to whom the user wishes to grant or deny access. This is because the security settings defined initially by the user will be used to determine which peers should gain access. The user only needs to define the security settings once, regardless of how many users are to be given access. When contrasted with access control lists, which requires the user to essentially examine each individual peer

and determine whether they should gain access before even posting the data onto the network, PDAC uses the security settings to determine access rights at access time, saving the user a considerable amount of effort.

PDAC also enforces original access constraints by using a data leak prevention mechanism which detects when a user is attempting to republish a data item that he or she has recently accessed under weaker confidentiality constraints. This will ensure that only peers who satisfy the security constraints actually gain access to data while all other peers do not.

In order to evaluate PDAC, simulations were performed in order to compare it with a hop-based scheme. The comparison was done by measuring the success rate of malicious users attempting to access protected data. An analysis of the user effort and processing time complexities of core functions will show that PDAC can work on large scale networks. Further, a PDAC prototype has been implemented on top of a social network emulator that faithfully represents existing social networks. These evaluations will demonstrate that PDAC is an effective scheme for use in existing social networks.

1.3 Topics Discussed

Before discussing PDAC in detail, we will first examine existing access control technologies in Chapter 2, where we discuss their strengths and weaknesses. In Chapter 3, requirements of a personal data access control scheme are discussed. An in-depth discussion of the PDAC scheme follows in Chapter 4, where the theory behind PDAC is fully explained. A security and complexity analysis of the scheme is performed in Chapter 5. The design of the PDAC prototype, which was written using the Ruby

Programming Language, is explained in Chapter 6. Results of simulations of the PDAC scheme is presented and discussed in Chapter 7. An overview of related work is presented in Chapter 8. Finally, directions of future work is dealt with in Chapter 9.

Chapter 2

Background Information

In this chapter, the term "social network" is defined, and we shall see how it can be modelled. A summary of the major access control paradigms follows, where their strengths and weaknesses are examined. Finally, we outline some of the requirements for access control systems with respect to collaborative systems.

2.1 Modelling Social Networks

Before we can delve into access control schemes, we must first define the term *social network*. A social network consists of a finite set of actors and the relations defined between them [3]. An actor can be a single person, or a group of persons. Actors in a social network are tied by relationships. The type and trust level of these relationships can vary depending on the actors involved. Examples of types of relationships include *friend*, *family*, or *coworker*. For the purposes of this thesis, we shall focus on trust of a relation. The trust level denotes the trustworthiness that a person assigns to a particular relationship. For example, family members may be more highly trusted

than a work colleague. Trust can be computed in a variety of ways, but is almost always computed as a function of the path between the source and target user [4]. Note that trust is directed; Bob may trust Alice, but Alice may not necessarily trust Bob.

A popular method of visualizing social networks is by means of a graph. Actors are represented by nodes and relationships are represented by directed edges. Relationships can be classified into two categories: direct and indirect [5]. Two actors have a direct relationship when an edge exists between them in the social network graph. An indirect relationship exists between actors a and b when there is no edge linking a and b, but there exists a path in the graph from a and c and a path from c to b. We refer to the least number of edges or "hops" between any two actors in the social network as the hop distance. A hop distance of 1 indicates a direct relationship, 2 hop distances indicate an indirect relationship (e.g. *friend-of-a-friend*), 3 hop distances indicate a *friend-of-a-friend-of-a-friend*, etc. It is important to note that the hop distance assumes that each relationship is equal and is separate from the trust level. We define the x-hop social neighbourhood of a user as all the users that are within x hops of that user. For the purposes of this thesis, we shall assume a 2-hop social neighbourhood unless otherwise specified.

2.2 Major Access Control Policies

There are three major Access Control policies that have emerged since the 1970s: Discretionary Access Control (DAC), Mandatory Access Control (MAC), and more recently, Role Based Access Control (RBAC). These access control policies are the among the most commonly used in computer systems.

2.2.1 Discretionary Access Control

Discretionary Access Control (DAC) [6], is an owner-centric policy where protected data is assigned ownership to certain entities. Most implemented policies are related to DAC in some form or another [7]. Three access control entities are defined: protected objects (those entities which are to be accessed), subjects (the entities which access the protected objects), and access rights (specifies operations that subjects are allowed to perform on objects). The owner of an object has complete control over which subjects are allowed access. Subjects are granted access only if the access rights authorize them. If the default behaviour of a DAC policy is to deny access, they are called closed policies. They can also be called "whitelisting" because they require an explicit specification of which subjects should gain access to the object. Open policies allow access by default, and require an explicit specification of subjects who should be denied access. These policies may also be referred to as "blacklisting".

Advantages of DAC is its simplicity, flexibility, and ease of implementation [7]. DAC policies have the drawback that access restrictions can be easily bypassed [2]. A subject who has been granted access can easily pass the object to non-authorized subjects without the owner's knowledge. This is because there is no restriction imposed upon the dissemination of information once a subject has gained access. DAC is usually implemented by means of Access Control Lists [2].

2.2.2 Mandatory Access Control

Mandatory Access Control (MAC) [8], protects data on the basis of security classifications of subjects and objects in the system. The security classification of a subject (or clearance) corresponds to the trustworthiness of that entity, while the security

classification of an object corresponds to the sensitivity of the information contained in the object. In contrast to DAC, object owners do not make access decisions [7]. Access is granted only if a subject has the necessary clearance in order to access an object. MAC is most often used in military applications, since it fits naturally with military policies. Security levels are hierarchically ordered and higher security levels are said to dominate lower security levels. A subject gains access if its security clearance dominates the security level of the object to be accessed.

An advantage of DAC over MAC is the fact that information flow is controlled. This is done by means of the following two principles [2]:

- **Read down**: A subject's clearance must dominate the security level of the object being read.
- **Write up**: A subject's clearance must be dominated by the security level of the object being written.

These two principles ensure that information does not flow from high level objects to low level objects. The one way flow of information is based on a lattice of security labels that is at the heart of MAC [9]. Implementing MAC usually requires some sort of centralized architecture. Typically, a security administrator is responsible for maintaining the security levels of subjects or objects. The rigidity of MAC makes it difficult to adapt it to a social network context.

2.2.3 Role-Based Access Control

Role-Based Access Control (RBAC) [10], has emerged in the past decade as the most widely discussed alternative to DAC and MAC. RBAC assigns permissions to well-defined abstractions called roles instead of to individual users. Roles can be defined as

a set of actions and responsibilities associated with a particular working activity [2]. Users then must take on different roles in order to gain access to protected objects. RBAC allows a user to activate and deactivate permissions by taking on a role in a given session [11]. A user who takes on a role inherits all access rights given to that role. Generally speaking, users can take on different roles on different sessions, and different users can take on the same role in simultaneous sessions.

RBAC provides easier management of access rights by breaking down the task of user authorizations into two parts: one part involves assigning access rights to a role and the other part involves assigning roles to users. This allows for simple assigning and revocation of access rights. RBAC also allows for a hierarchy of roles, making it an ideal fit for many applications with roles that can be highly specialized. However, context in the activation, deactivation, and management of roles are not fully considered. Also, in collaborative environments, a user in an instance of a role might need a specific permission on only one instance of an object and RBAC does not have the ability to specify control that is fine-grained enough to deal with this case.

2.3 Other Access Control Models

The access control policies discussed in Section 2.2 are the most widely used policies today. However, a few other access control models have been developed that attempt to address a few of the weaknesses of the big three policies. Most of these models are extensions of RBAC and have been proposed or used for collaborative environments.

2.3.1 Task-Based Access Control

Task-Based Access Control (TBAC) [12] extends the traditional subject/object-based access control models by granting permissions in steps related to the progression of tasks. Permissions are associated with a protection state. This allows for type-based, instance, and usage-based access, and gives authorizations an expiration date. Weaknesses include having race conditions due to the "just-in-time" activation and deactivation of permissions, and the primitive nature of the specification of complex policies and management of authorization policies. A collaborative framework that uses TBAC in conjunction with RBAC has been presented in [13].

2.3.2 Team-Based Access Control

In most organizations, individuals are usually divided into teams which are then used to associate a collaboration context with an activity. Team-Based Access Control (TMAC) [14] defines two parts of the collaboration context. User context allows identification of specific users that are taking on a role at any given moment, while object context identifies specific objects that are required for collaboration. In addition to the strengths of RBAC, this model allows for fine-grained control on individual users in certain roles and on individual object instances. However, the TMAC model is not yet fully developed, and does not fully reflect the multidimensional nature of collaborative contexts such as organizational entities, workflow tasks, and groupware's environmental components [15].

2.3.3 Spatial Access Control

Spatial Access Control (SAC) [16] considers the environment of collaboration and uses it to hide explicit security mechanisms from users. It was developed primarily for collaborative virtual environments. One component of the model called a boundary divides the collaborative environment into regions and uses credentials to access these regions. The second component called an access graph specifies movement constraints in the collaboration space and manages the access requirements. However, this model does not provide fine-grained access control, and requires an application domain that can be divided into regions and boundaries.

2.3.4 Context-Aware Access Control

Context-Aware Access Control [17] is an extension of RBAC that activates roles based on environmental conditions at the time of request. Role hierarchy, activation, and separation are used to manage policies and constraints that depend upon the context of collaboration. This model is useful in applications where environment-sensitive information is readily available. However, this model has not been fully tested in the collaborative systems domain.

2.4 Access Control Requirements in Collaborative Systems

The access control policies discussed in Section 2.2 were initially designed for single-user environments. The models in Section 2.3 have attempted to address several requirements which are pertinent to collaborative systems. Tolone, Ahn *et al.* [15] performed a comprehensive study of various access control models and identified key requirements for access control in a collaborative environment. One such requirement

is that access control must be applied and enforced at a distributed platform level. Another requirement is to have a generic model that is expressive enough to efficiently specify access rights based on various information. This means that access control models need to be generic and configurable in order to meet the requirements of cooperative tasks and enterprise models. Greater scalability in terms of the number of user operations is also essential because of the higher number of shared operations in collaborative environments.

In addition, access control models must also allow for varying levels of granularity for the protection of resources and data, while at the same time allow a high level specification of this protection [15]. Access control models must also maintain a strong exclusion of unauthorized users in a manner that is flexible enough so as to not constrain collaboration. Access control models must also be dynamic, allowing policy changes at runtime depending on the environment. Access administration is also an important requirement for access control models in collaborative environments. Access control systems must allow for fine-grained protection, administrator assignment, ownership issues, and the delegation and revocation of authorizations [18]. Finally, there should be reasonable bounds on performance and resource costs.

Chapter 3

Protecting Personal Data in Social Networks

In this chapter, we discuss requirements that are desirable in a personal data protection scheme. Note that these requirements are specific to a social networking context. We begin by illustrating a simple protection scheme and discuss the issues arising from using such a scheme.

3.1 Simple Protection Scheme

The structure of social networks can be exploited to create an access control scheme for personal data. A simple example of this is a scheme based on hop distance. Suppose Alice wants to show her friends some personal photo albums. In order to do this, she uploads the data onto a social networking website. She specifies that only her immediate friends will be able to view the album. In a social networking context, this means that only those peers which are one hop away should be able to view the album.

In the social graph depicted in Figure 3.1, only users Bob, Carol, Frank, and Kate would be allowed access to Alice's data. If Alice were to publish data that she wishes a larger group of people to access, say for example, a poster indicating a garage sale, she could specify that the friends of her friends should be allowed access. In a social networking context, this means that peers which are within two hops away should be able to view the poster. This would correspond to David, George, Eve, Joyce, and Nancy in Figure 3.1. This scheme could be extended to as many hops as deemed necessary by Alice.

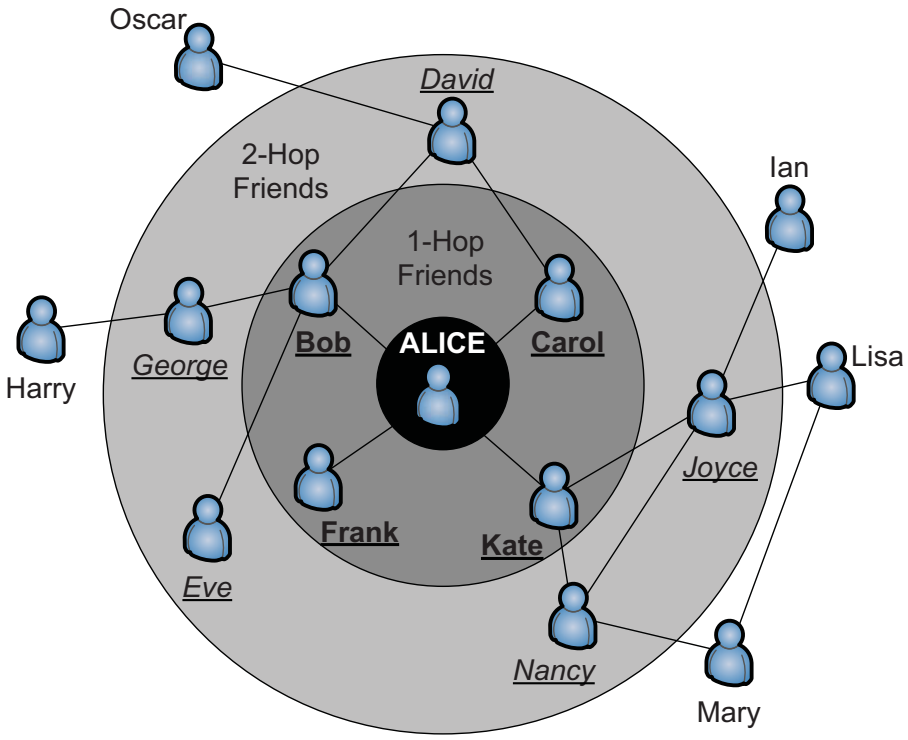

Figure 3.1 Example social graph. Alice's 1-hop friends (in the dark gray area) are Bob, Carol, Frank, and Kate. Her 2-hop friends (in the light gray area) consist of David, Eve, George, Joyce, and Nancy. Her 2-hop social neighbourhood consists of all her 1-hop friends and 2-hop friends.

This hop-based scheme is simple to understand and easy to implement. Current social networks such as `facebook.com` and `myspace.com` already employ this simple access control scheme. However, a key drawback of this scheme is that it does not allow for granting only specific peers access to a published resource. This can be remedied by the use of access control lists to specify who should and should not be granted access to the resource. `Facebook.com` uses this approach by implementing blacklists and whitelists when specifying the privacy settings of a particular photo or album. The "network" feature of `facebook.com` also allows users to limit resource access only to users who belong to a specific organization or who reside in a specific geographic area.

While the simplicity of the above scheme is intuitively appealing, it has some major drawbacks. Many of these drawbacks stem from the following assumptions made by the simple schemes that are not always applicable:

(i) **All friends are equal:** Access control schemes that categorize friends according to hop distance assume that all friends at a particular hop distance are equal. While this simplifies access control, resource owners often need the flexibility of differentiating among friends for certain data objects. Data objects of a more personal nature may require the owner to be more selective among his or her friends.

(ii) **No impact on friendships:** The simple hop-based protection scheme assumes that access control decisions do not impact the topology of the social network – which is often not valid. If, for example, Alice is unwilling to provide access to Carol for data that she is already sharing with Bob, then it can indicate a lack of trust on Carol.

(iii) **Omniscient friends:** Access control schemes that expect users to define access control lists to explicitly deny and allow accesses to data objects assume that users are all knowing about their social neighbourhood and able to make the appropriate protection decisions. With large and dynamic social neighbourhoods (friends and friend-of-friends), such an assumption is not practical.

(iv) **Friendships are static:** Because the nature of friendships can change from time to time, this can impact the owner's desire for certain users to access his or her data at certain times. In order to deal with this scenario, access control schemes utilizing only hop distance would require breaking and restoring the friendship link. Access control lists would require users to add and remove specific peers from each list, which may be numerous.

One more issue remains to be addressed. Like DAC (see Chapter 2), this simple protection scheme does not prevent further dissemination of the protected data after authorized users have been given access. Once users gain authorized access, it is incumbent upon them to ensure that they do not then give access to other users who would otherwise have been denied access. The original data owner has no real control over the spread of the data once other authorized users have gained access.

3.2 Requirements of Personal Data Protection

A scheme to protect personal data should provide the user with full control over how his or her data is shared over the social network. A person's data collection can be incredibly diverse, ranging from photo albums to important documents to home made videos. Depending on the content of the data, sharing requirements can be just as

diverse. Some data may be considered more personal than others and so is meant for only a select group of friends. Other data may require a larger audience to be reached. The criteria for choosing which peers should gain access must be directly proportional to the confidentiality level of the data. Therefore, a personal data protection scheme must be fine-grained enough to ensure only highly trusted peers access sensitive data.

In addition, the scheme must be flexible in other ways, such as being able to apply different sharing criteria for different friends. It must also be flexible enough to deal with changing sharing conditions on a data object basis (for example, increasing the confidentiality level of an already published data object), as well as on a friend-by-friend basis (for example, decreasing the trustworthiness of a friend and consequently limiting his or her access to the owner's data). While both flexibility and granularity are essential in protection schemes, they can add significant overhead in terms of user effort required to setup and maintain a protection regimen.

The user effort required by the data protection scheme is certainly a major factor that must be considered. One way of retaining the flexibility and reducing the user effort required by the scheme is to make use of previous access control decisions. This historical information can come from different sources:

(i) **Personal History:** Learning from the past actions of the resource owner with respect to access control.

(ii) **Social Neighbourhood History:** Learning from the past actions of the community of users within the owner's social neighbourhood.

(iii) **Like-Minded Peer History:** Learning from the past actions of a like-minded set of users within the owner's social neighbourhood.

A combination of the above sources can be used to minimize the amount of user effort required. Sources (ii) and (iii) are especially interesting as they both enable collaborative decision making that exploits the structure of the underlying social network. Collaborative decision making allows untrustworthy peers to be identified more quickly than through ordinary direct interactions.

Specifying confidentiality constraints is but one part of the objectives of a personal data protection scheme. These confidentiality constraints must then be enforced. The protection mechanisms should allow the users to specify and enforce confidentiality policies that are tailored for the different data objects. Two concerns with regard to confidentiality constraints must be dealt with: initial access to data objects and reposting of data objects. A personal data protection scheme must address these concerns in a consistent manner, such that a copy of a previously posted data object is subject to the same initial access constraints as the original with respect to the original owner.

3.3 Modelling Trust

Trust is one of the important measures derived from the structure of the social network and the activities of the users in the social network. To keep the overall technique simple, we need a trust model that effectively captures the notion of trust between users, is intuitively appealing, and simple to implement. To quantify the trust between two users, a *trusted distance* measure is required. The trusted distance measure should be based in part on the hop distance between users on current online social networks. In this model, the trusted distance measure would be inversely proportional to the actual trustworthiness of a user. A low trusted distance value between two users

indicates a high degree of trust, whereas a high distance value indicates a low degree of trust. Unlike simple hop distance, the value of the trusted distance measure is dependent upon its direction. For example, in the social graph in Figure 3.1, the trusted distance value from Alice to George may not necessarily be the same as the trusted distance value from George to Alice, despite the fact that their hop distances are identical. Unlike the integer values of hop distance, the trusted distance is a real value. Also, trusted distance can include hop distance along with other factors relevant to inter-personal trust such as interaction history.

3.4 System Assumptions

For this thesis, the following assumptions are made. We assume all users belong to a centrally maintained social network that is connected and structured as a single giant component. There are no "super-users" who are directly connected to everyone in the social network. The interconnections among users on the social network are context independent (e.g., direct friends of Alice could include office colleagues and family friends). The social network follows the best security practices to prevent the theft of user credentials. Finally, we assume that most of a peer's friends in the social network are genuine acquaintances with a vested interest in keeping their social neighbourhood safe from the relatively small set of malicious users in the social network.

The access control scheme such as the one presented in this thesis could be guarding the data posted on the social network (e.g., profile information such as address, date-of-birth, personal photograph) or personal data stored on online storage systems. In either case, the objective of the access control scheme is to use trust measures derived from the social network to control the data sharing activities.

Chapter 4

Personal Data Access Control

In this chapter, we discuss in detail the *Personal Data Access Control* scheme, or PDAC. PDAC is designed to protect personal data that users post on social networks and other online storage systems. When a user or *publisher* posts a data object, he or she specifies the confidentiality constraints using a trusted distance measure. As will be discussed in Section 4.2, the confidentiality specifications divide the people seeking access into three zones with respect to the publisher: highly trusted friends, marginally trusted friends, and untrusted people. Requests from highly trusted friends are automatically honoured while the requests from marginally trusted friends are honoured only if the attesters designated by the publisher are willing to undersign the trustworthiness of the requester. Requests from untrusted people are automatically rejected.

4.1 Data Storage

PDAC relies upon the availability of a centralized *trusted data store* (TDS) that can be used by all users to store and retrieve data objects. The TDS is composed of four separate components: *Content Manager*, *Access Request Manager*, *Trust Manager*, and *Data Leak Manager*. Users post their data objects on the TDS, where it is stored at the Content Manager along with the desired confidentiality constraints. The Data Leak Manager analyzes the data objects to determine if an information leak has occurred, and adjusts the confidentiality constraints if necessary (See Section 4.8). The Access Request Manager is responsible for enforcing the confidentiality requirements of the data objects stored at the Content Manager. In addition, the Access Request Manager logs access requests and outcomes of the requests (that is, whether access requests were accepted or rejected). This activity log is used by the Trust Manager in the trust computation process as described in Section 4.3. In Section 4.9 this scheme is extended to use multiple instances of TDS to provide fault tolerance to the system.

4.2 Confidentiality Limits and Specification

Confidentiality requirements in PDAC are specified as follows. For each data object that is published, two threshold values are defined, designating three distinct data protection zones with respect to the publisher. An access request for data object *obj* owned by publisher P is automatically granted if the requester's trusted distance from P is less than the *accept limit*, $c_{acc}(P, obj)$. Peers whose access requests are automatically granted are said to fall in the *Acceptance Zone* for that data object. The *reject limit*, $c_{rej}(P, obj)$, is the smallest trusted distance from P for which an

access request is automatically rejected. Peers whose access requests are automatically rejected are said to fall in the *Rejection Zone* for that data object. Note that the limits satisfy $0 \leq c_{acc}(P, obj) \leq c_{rej}(P, obj)$. If the trusted distance from P to a requesting peer is in the semi-closed interval $\big[c_{acc}(P, obj), c_{rej}(P, obj)\big)$, the request for access requires authorization by attesters designated by P. Peers whose access requests require attestation are said to fall in the *Attestation Zone* for that object.

Confidentiality is enforced by computing the trusted distance from the publishing peer to a requesting peer and using that value to map the requesting peer in the appropriate protection zone for the data object in question. The PDAC protocol employs a cooperative strategy in computing the trusted distances. As shown in Section 4.3, trust measures are computed by using information from prior user activities.

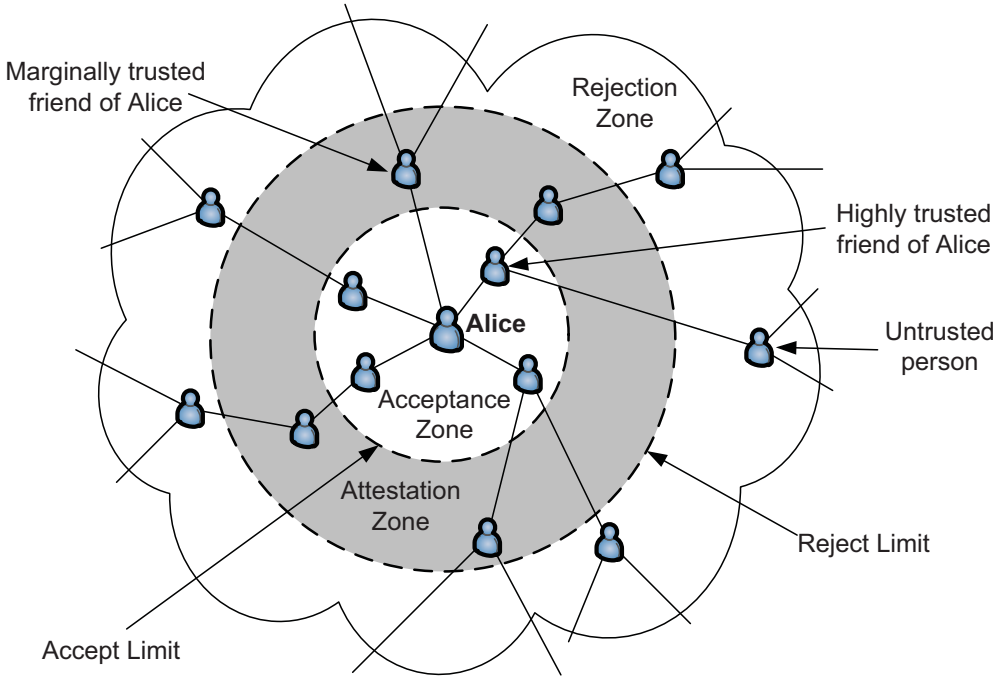

Figure 4.1 Data Protection Zones. Illustration of data protection zones for one of Alice's data collections with user mappings.

4.3 Trusted Distance Computation

The hop distance (see Chapter 2) is a fairly static parameter that only changes when new relations are added or old ones removed. To keep the social graph simple, the relations are usually context independent (i.e., work and family friends are linked the same way in most social networks). To capture the historical context introduced by interactions between any two users x and y in the social network, we define a measure called *affine distance* – $d_{aff}(x, y)$, which is computed by mining past user activities. For the purposes of this thesis, we assume that $d_{aff}(x, y)$ is dependent upon how x or the users in x's social neighbourhood have honoured y's access requests in the past. We say that $d_{aff}(x, y)$ is the affine distance from x to y.

4.3.1 Social Neighbourhood Success Rate

The Access Request Manager of the TDS maintains a log of all user activity for a pre-defined time window (e.g., all access requests within the past week or month). The activity log for user y maintains information on all the requests y made within the time window and the associated outcomes. Each log entry stores the id of the data object requested by y, the identity of the original poster of the data object, the time of request, tags describing context, and the outcome of the request. Suppose y's activity log has $q_y(x)$ total requests to users in x's social neighbourhood over the current activity window. Of the $q_y(x)$ requests, suppose $a_y(x)$ were accepted and $r_y(x)$ were rejected such that $q_y(x) = a_y(x) + r_y(x)$. This historical information is compressed into a value called the social neighbourhood success rate, $s_y(x)$, by computing an aggregate success rate. The raw success rate is scaled by a factor which depends upon the unique number of data posters p that have accepted requests made by y. The

scaling factor is such that a user whose access requests have been accepted by a large number of unique data posters will have a higher social neighbourhood success rate than another user with the same raw success rate but whose access requests have been accepted by only a few other users. The scaling factor is essentially a credibility score for the raw success rate, and is used to thwart users from colluding with a small set of other users. Intuitively, the social neighbourhood success rate $s_y(x)$ is a measure of how trustworthy y is deemed to be by x's social neighbourhood. The value of $s_y(x)$ is in the interval $(-1, 1)$, and is computed as follows:

$$s_y(x) = \begin{cases} 0 & \text{if } q_y = 0, \\ (\frac{r_y - a_y}{q_y})(\frac{1}{1+e^{\beta - p/\alpha}}) & \text{if } q_y \neq 0. \end{cases} \tag{4.1}$$

In Equation 4.1, α and β are system parameters used to control how quickly the scaling factor approaches 1 as p increases. If they are set too low, a high credibility score for the success rate can be obtained much too easily with only a few unique data posters. If set too high, obtaining a high credibility score will be much too hard, requiring a high number of unique data posters. The user is free to change the values of these parameters, but the system can assign default values that should suffice for most users. A positive $s_y(x)$ value indicates that most of y's requests to x's social neighbourhood have been rejected, while a negative value indicates that most of those requests were accepted.

4.3.2 Affine and Friend Distances

The value of the affine distance $d_{aff}(x, y)$ can be computed using the information in the activity log of y. Let q_{yx} be the total number of requests y has made for x's

data objects. Of these requests, let a_{yx} be the number of accepted requests and r_{yx} be the number of rejected requests such that $q_{yx} = a_{yx} + r_{yx}$. Let λ be a tuning parameter in the interval $[0, 1]$ that controls the proportion of the contribution of y's social neighbourhood success rate to the affine distance. Let Δ be an arbitrary small number. To obtain $d_{aff}(x, y)$, the social neighbourhood success rate of y given by $s_y(x)$ is combined with the success rate of y for x's data objects (called the individual success rate) as shown in the following equation:

$$d_{aff}(x, y) = \lambda s_y(x) + (1 - \lambda)(\frac{r_{yx} - a_{yx}}{q_{yx} + \Delta}) \tag{4.2}$$

The λ value essentially allows user x to control the effect of his or her social neighbourhood's trust in y, as indicated by $s_y(x)$, on the affine distance from x to y. A higher λ value will result in the affine distance being significantly affected by y's reputation in x's social neighbourhood, while a lower value places more importance on x's personal impression of y based upon the individual interactions between them. Typically, default values would be chosen such that $0.3 \leq \lambda \leq 0.5$ so as to ensure that personal interactions are weighted more than social neighbourhood interactions, but the user is free to change the value to suit personal preferences. The Δ value is is a system defined parameter added to q_{yx} in Equation 4.2 to ensure that $d_{aff}(x, y)$ is in the range $(-1, 1)$. Affine distances closer to -1 indicate a high degree of trust based on previous interactions, while distances closer to 1 indicate a low degree of trust.

While the hop and affine distances are useful for indicating trust between users, they are inadequate for dealing with sudden changes in protection requirements. To make rapid changes in protection requirements, we use two *friend distance* parameters: an *all-friends distance* and *per-friend distance*. The all-friends distance, $d_{afdst}(x) \geq 0$,

is applied between user x and all other users. This parameter has a default value of 0. A non-zero value indicates that user x wants restricted sharing for all data that he or she owns. Similarly, the per-friend distance, $d_{pfdst}(x, y) \geq 0$, is applied between user x and user y and is also set to 0 by default. A non-zero value here indicates that user x wants to restrict data sharing with user y. The higher the values for $d_{pfdst}(x, y)$ and $d_{afdst}(x)$, the more severe the restriction for y and all requesters, respectively. The values of both friend distances are user-defined.

4.3.3 Trusted Distance

The friend distances $d_{afdst}(x)$ and $d_{pfdst}(x, y)$, along with the hop distance $d_{hop}(x, y)$, are added to the computed affine distance $d_{aff}(x, y)$ between x and y to obtain the trusted distance $d_{trust}(x, y)$:

$$d_{trust}(x, y) = d_{hop}(x, y) + d_{aff}(x, y) + d_{afdst}(x) + d_{pfdst}(x, y) \tag{4.3}$$

A low $d_{trust}(x, y)$ value indicates x's high degree of trust in y. Conversely, a high value indicates x's lack of trust in y. Note that in equation 4.3, $d_{hop}(x, y)$ uses positive integer values such as 1 and 2 to represent a friend and a friend-of-a-friend, respectively. The $d_{aff}(x, y)$ value is in the range $(-1, 1)$. The friend distances $d_{afdst}(x)$ and $d_{pfdst}(x, y)$ are both nonnegative. This means the trusted distance of any peer can only decrease by a value strictly less than one hop count. In other words, the computed trust of an indirect friend (friend-of-a-friend) will never be less than the hop distance of a direct friend. A trusted direct friend (indicated by $d_{aff} < 0$ and $d_{pfdst} = 0$) will always be considered more trusted than a trusted friend-of-a-friend by this scheme.

4.4 Zone Classification

The value of $d_{trust}(P, R)$ determines which zone a requesting peer R belongs to with respect to a publishing peer P. Taken together with the confidentiality limits set by P on his or her data objects, the trusted distance determines whether data object *obj* belonging to P is accessible to requester R. Users are classified into the zones by means of the following rules:

1. $d_{trust}(P, R) < c_{acc}(P, obj)$**:** User R is considered highly trusted by P and is automatically given access to the data object. This is called the *Acceptance Zone.*

2. $c_{acc}(P, obj) \leq d_{trust}(P, R) < c_{rej}(P, obj)$**:** User R must be *attested* to access the data object. This is called the *Attestation Zone.*

3. $c_{rej}(P, obj) \leq d_{trust}(P, R)$**:** User R is automatically denied access to the data object. This is called the *Rejection Zone.*

For Cases 1 and 3, no further processing is required. Case 2 will trigger the attestation procedure described in Section 4.5. Note that because a user's activity log maintained at the Access Request Manager is affected by which data objects he or she has been given access to, a user can conceivably move from one zone to another. A series of accepted access requests can push a user from the Attestation Zone into the Acceptance Zone, while a series of rejected requests can easily push a user into the Rejection Zone.

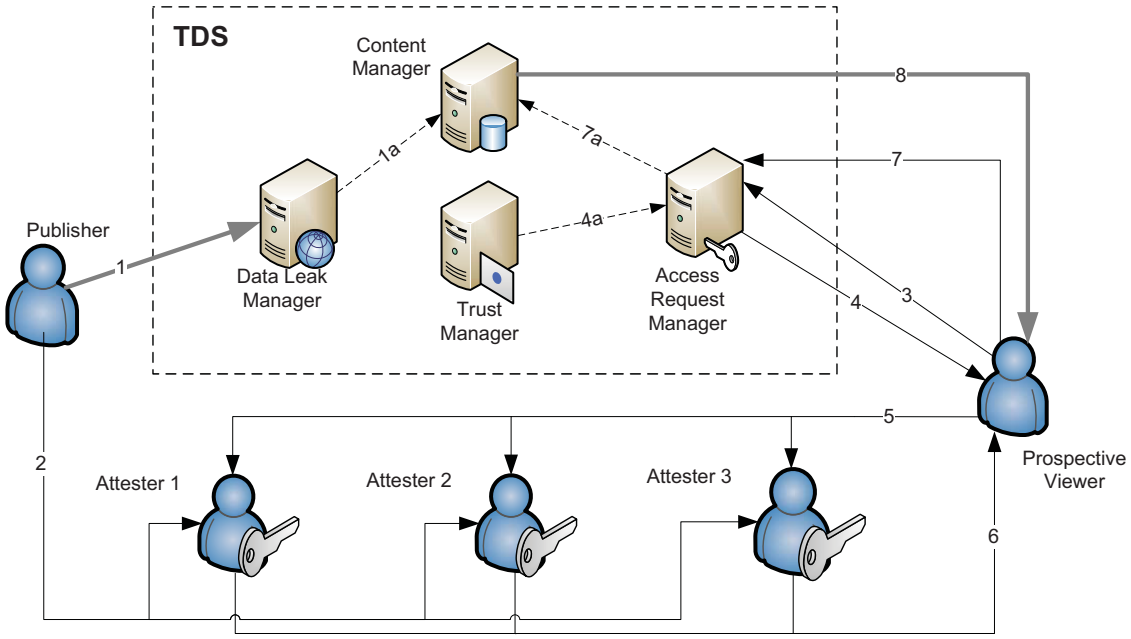

Figure 4.2 The steps involved in publishing and requesting data using PDAC. To publish a data object, the publisher sends it to the TDS (1) and notifies the attesters (2). Before accepting the data object for publication, the TDS performs a data leak analysis and adjusts the confidentiality limits if needed (1a). A prospective viewer requests TDS to access the data (3), and receives either an accept notification, reject notification, or RFA certificate from TDS (4a, 4) which needs to be signed by the attesters(5). If an RFA certificate is given, the prospective viewer must obtain the necessary signatures (6). The attested RFA certificate is submitted to the TDS (7), and if the certificate is valid (7a), the viewer is granted access (8).

4.5 Attestation Process

In addition to the confidentiality limits, a user seeking to publish a data object must specify a list of n attesters and a value k indicating the minimum number of attester authorizations required to gain access to the data. The attesters are members of the social network whom the publisher considers trustworthy. When a requesting user falls into the Attestation Zone, the attesters are asked to vouch for the trustworthiness of the requester. If at least k out of n attesters can certify that the requester is trustworthy enough to gain access to the data, the Access Request Manager will grant the requester access.

The criteria that the attesters use to determine whether a requester is trustworthy enough to access the data is specified by the publisher. Typically, the criteria will specify that the requester be within a certain hop distance of the attester for the attester to sign off on the request. For example, Alice can specify that the attesters should limit access authorizations to requesters that are within 2 hops of them. Affine distances are not used in order to prevent collusion among the attesters. The choice of attesters for any particular data object is at the discretion of the publisher. For example, Alice could enlist family members whose judgment she trusts as attesters when she wants to protect data objects such as a family photo album. However, those same attesters may not be appropriate for a business document authored by Alice, so she would instead enlist trustworthy co-workers as attesters.

To ensure the security of the attestation process, a *request for attestation* (RFA) certificate may be used. When a user requests a protected data object from the TDS, the Access Request Manager assigns the user an RFA certificate that lists the n attesters and a value for k. Once the user successfully obtains at least k attestations on

the RFA certificate, it is submitted to the Access Request Manager. The attestations should include a digest of the RFA certificate to prevent reuse of old attestations. The RFA certificate can include an expiry time to force the client to complete the attestation process within a limited time frame.

4.6 Sharing Pseudo-Blacklists

PDAC can emulate blacklist functionality by using an infinite per-friend distance. A further extension to the blacklisting functionality would be to share the per-friend distances among a user's 1-hop friends. This allows the user to take advantage of his or her friends' knowledge of malicious users in the network. This will also act as a deterrent to malicious users, since sharing blacklists ensures that news of their malicious behaviour will spread quickly throughout the social neighbourhood. The more quickly publishers learn of other users' malicious behaviour, the better they can protect their data object by assigning a high per-friend distance to those malicious users. However, there may be instances where a user may consider a friend trustworthy but the rest of his or her peers may consider that user malicious. To prevent a user's personal security settings from being overridden by that of his or her immediate friends, a user's per-friend distance must take precedence over the per-friend distances provided by his or her friends. Thus, the user can take advantage of the experience of his or her friends while maintaining control over his or her personal security settings.

4.7 Changing Access Limits

Because data protection concerns can change over time, it is important to provide the user with the flexibility of changing the access control parameters even after posting the data objects. Changes to the security parameters supported by PDAC include:

1. **Modifying access control parameters for a single data object:** A publisher may want to modify access control parameters for a particular data object that he or she has previously posted on the TDS. With the original posting, the publisher specified two thresholds that correspond to accept and reject limits. These values can be changed at the Content Manager to reflect the new access control requirements.

2. **Modifying attestation criteria for data objects:** The attesters examine requests from users that have a trusted distance that falls between the two limits. When a data object is published on the TDS, the publisher associates attestation criteria with it. The criteria are stored with the data object at the Content Manager in the TDS and are noted down in the RFA certificate issued by the Access Request Manager. To change the attestation criteria, the publisher can retrieve the records associated with the data object in the Content Manager. For example, Alice could specify that attesters should verify that the requester is within 1 hop instead of 2 hops.

3. **Restricting access for all data objects:** A publisher may want to restrict access to all data objects that he or she has published on the TDS. This can be easily accomplished by increasing the all-friends distance d_{afdst}.

4. **Restrict access for a particular user:** A publisher P may want to prevent a given user R from accessing some of his or her data objects. Such a restriction can be achieved by increasing the per-friend distance that corresponds to R, $d_{pfdst}(P, R)$.

4.8 Data Reposting and Information Leakage

The PDAC protocol described so far deals with the original posting of a data object. To maintain the confidentiality of a data object, reposting of data objects must also be considered. In particular, reposts that are based on previous content held by the Content Manager in the TDS should have confidentiality limits that are consistent with the confidentiality requirements of the original data object. For example, suppose Alice posts a personal photo album on the TDS with reject limit of 2. A friend of Alice, Kate, can access it and repost it with her comments. The confidentiality limits specified as part of the original data (Alice's photo album) does not restrict the confidentiality constraints Kate uses. Even if Kate were to repost the same album with a reject limit of 2, users who do not meet the original trusted distance requirement can still access Alice's photos. This is because the trusted distance of requesters seeking to access Kate's repost of Alice's photo album will be computed with respect to Kate. Assume, for simplicity, zero friend and affine distances between Joyce, Alice, and Kate. Joyce should be denied access to Alice's posting, since the hop distance from Alice to Joyce is $d_{hop}(A, J) = 2$. However, since the hop distance from Kate to Joyce is $d_{hop}(K, J) = 1$, Joyce can access Alice's photo album through Kate's reposting. This clearly violates the original confidentiality requirements.

4.8.1 Detecting Information Leaks

To prevent such inconsistent confidentiality specifications that can lead to information leakage, the TDS has a Data Leak Manager which enforces constraints on the confidentiality requirements associated with a data object. For instance, when Kate submits a data object to the TDS for posting, the Data Leak Manager performs a similarity analysis of that object against the data objects to which Kate recently gained access. In this context, recently accessed objects are those items that Kate has gained access to within the activity log's time window. If the data object presented by Kate is sufficiently different from those data objects that she recently accessed, the data object is accepted for posting. Otherwise, its confidentiality specifications must be modified to be consistent with the original constraints.

A data object is considered sufficiently different if a computed similarity measure is less than a pre-defined threshold τ. For each data object *obj* posted on the Content Manager, the Data Leak Manager extracts feature vectors using document similarity measurement techniques. Data object similarity computation is beyond the scope of this thesis, but a large variety of different fingerprinting algorithms exist for documents [19, 20, 21, 22], images [23, 24, 25], audio files [26, 27] and video files [28]. A combination of these algorithms can be used to determine data object similarity. Using the feature vectors extracted by the Data Leak Manager for each data object, it computes a feature match score $0 \leq m_{ij} \leq 1$ between two objects obj_i and obj_j as the fraction of matching features.

Consider the case where user P is submitting data object obj_{copy} to TDS. Let $\mathcal{D}$ be the set of all objects stored on the TDS. Using access logs from the Access Request Manager, the Data Leak Manager determines the set $\mathcal{D}_P \subseteq \mathcal{D}$ containing the objects

recently accessed by user P. A similarity analysis by the Data Leak Manager of data object obj_{copy} against the set $\mathcal{D}_P$ reveals a set of data objects $\mathcal{D}_{Pm} \subseteq \mathcal{D}_P$ such that for each $obj_i \in \mathcal{D}_{Pm}$ we have a feature match score m_{Pi} that is above the similarity threshold τ used by the Data Leak Manager. One can reasonably conclude that the similarity (indicated by a high feature match score) between the object posted by P and the objects in $\mathcal{D}_{Pm}$ indicates that obj_{copy} is either a complete or partial copy of at least one of the objects in $\mathcal{D}_{Pm}$.

4.8.2 Time Window Size

One criticism of this leakage detection system is that a user can obtain access to a data object, wait until the access request is purged from that user's activity log, and then repost the data object. This can be addressed by simply increasing the size of the time window over which the activity log records data. The rationale behind this design is that for most data items, the relevance and level of interest in the data item decreases as the data item ages. Therefore, an appropriate time window size should be chosen such that by the time the original access entry has been purged from the activity log, either the social network as a whole has lost interest in accessing the data, or the user attempting to repost the data has lost interest in publishing it. Within a social networking context, this should be sufficient for most data items posted. However, since a large time window directly affects the size of the activity log, a tradeoff exists between the protection against information leaks and the memory and time complexity of the similarity analysis. This must be kept in mind when choosing the time window size.

4.8.3 *Strict* vs. *Relaxed* Dissemination

There is one more security parameter that a user seeking to publish data on the TDS can set. PDAC allows a publisher to choose between two dissemination settings: *relaxed* and *strict*. This parameter gives the user control over how far his or her data may be spread. Suppose Q is the owner of a matching data object $obj_{orig} \in \mathcal{D}_{Pm}$ and has specified $c_{acc}(Q, obj_{orig})$ and $c_{rej}(Q, obj_{orig})$ as the accept and reject limits respectively. The trusted distance from Q to P is $d_{trust}(Q, P)$ and the hop distance between them on the social graph is $d_{hop}(Q, P)$. If Q selected *relaxed* dissemination, P can only specify the accept limit such that $c_{acc}(P, obj_{copy}) \leq c_{acc}(Q, obj_{orig}) - d_{trust}(Q, P)$. Similarly, the reject limit is constrained by the inequality $c_{rej}(P, obj_{copy}) \leq c_{rej}(Q, obj_{orig}) - d_{trust}(Q, P)$. With a *strict* privacy setting, P's accept limit is constrained by $c_{acc}(P, obj_{copy}) \leq c_{acc}(Q, obj_{orig}) - d_{hop}(Q, P)$ and the reject limit is constrained by $c_{rej}(P, obj_{copy}) \leq c_{rej}(Q, obj_{orig}) - d_{hop}(Q, P)$. The *strict* setting will ensure that the data object does not spread further than $c_{rej}(Q, obj_{orig})$ hop counts from the original publisher Q. The *relaxed* setting allows the data to propagate but using only trusted paths in the social network.

4.9 A Distributed Trusted Data Store

We now discuss a distributed architecture for the Trusted Data Store, which will provide the system with fault tolerance. Each peer in the social network will be assigned to a "home" TDS server, through which all data store transactions for that peer will be processed. All data objects published by a peer will be stored at that peer's home TDS, and is assigned a data id that uniquely identifies the TDS which stores that data object. A peer's home TDS will also act as a gateway to other TDS

servers that store other peers' data.

When a user P submits a data object to the TDS for publication, P's home TDS will store the data object with his or her home Content Manager. This copy will act as the master copy of that data object. When another user R requests to see the data object, that user's home Access Request Manager will prompt the corresponding home Content Manager to retrieve the meta-data associated with that data object. The home Content Manager ascertains which TDS contains the requested data object. If a copy of the meta-data for the data object exists locally (i.e. on R's home TDS), the Content Manager will simply retrieve the meta-data from the local copy. If the data object is hosted on a remote TDS, the R's home Content Manager will contact the remote Content Manager and retrieve the meta-data for the desired data object. User R's home Content Manager will then store this information locally for future access. If R's home Content Manager already contains a cached copy of the meta-data, P's Content Manager is contacted in order to determine whether the cached version contains the latest copy. If the remote server contains a newer version of the meta-data, the local copy is overwritten by the remote copy.

Once the home Access Request Manager determines that the requester R is permitted to access the data object, the home Content Manager will once again contact the remote Content Manager and retrieve the actual data object. A local copy of the data object is stored on R's home Content Manager. Similar to the meta-data, if a locally cached copy of the data object exists, it is overwritten by the remote copy if it is newer. This method of retrieving the data will ensure that the data object is given to R's home TDS only if R is actually granted access. The requester R is then able to view P's data object by means of the local copy on his or her home TDS. In order

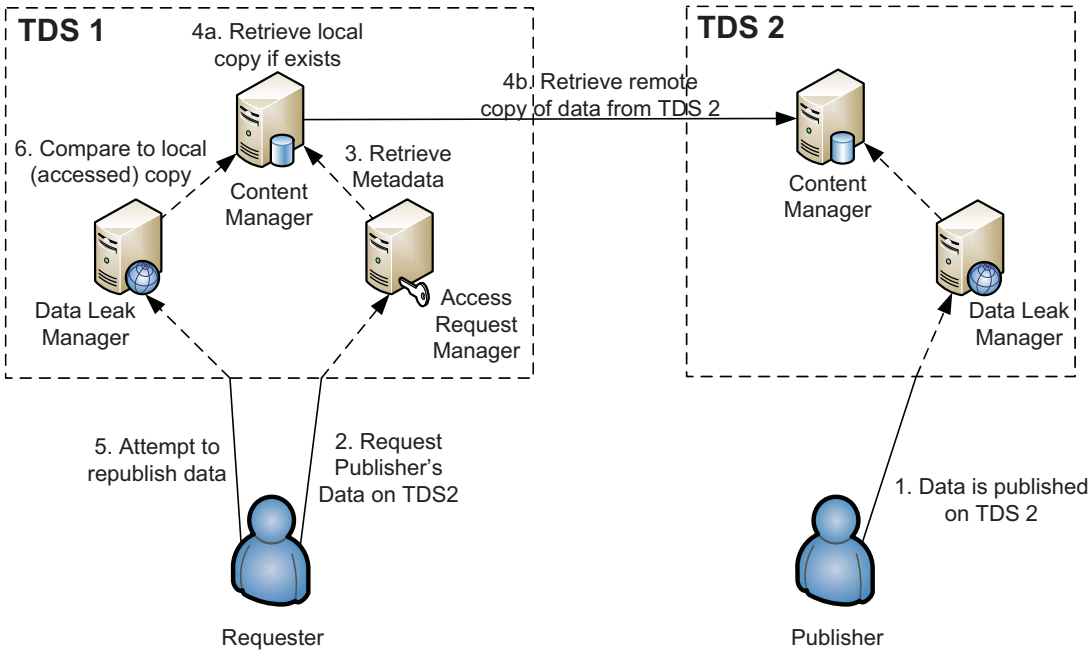

Figure 4.3 A distributed architecture for the TDS.

to conserve disk space, data objects which have not been accessed for a long time are purged from the cache.

If R attempts to repost P's data object under a weaker confidentiality, R's home Data Leak Manager will compare that data object with the local copy of P's data object. Since the only way that R would have seen P's data object would have been through the local copy, this will ensure that the comparison is performed against the copy to which R actually had access. Assuming changes to data objects published on the TDS are infrequent, this will also ensure that any changes that P makes to the data object after R accesses it will not be considered in the comparison.

Note that these transactions are all transparent to the user. From the user's viewpoint, there is only one TDS with which he or she interacts.

4.10 Example Scenario

Consider the following scenario, which uses the example social graph in Figure 3.1. Assume $\lambda = 0.4$ and $\Delta = 0.001$. Alice wishes to share an album of personal photographs *alb* with her friends using a Social Network. She wants her immediate friends as well as close friends of her immediate friends to be able to view these photographs. She publishes the photographs onto the TDS using a *strict* privacy setting with an accept limit of $c_{acc}(A, alb) = 0.5$ and a reject limit of $c_{rej}(A, alb) = 2.5$. She also assigns her friends Bob, Carol, Frank, and Kate as attesters for her photographs, specifying that requesters must obtain the authorization of at least 2 attesters and that attesters should limit authorizations to peers that are within 2 hops. David, a mutual friend of Bob and Carol, sends a request to the Access Request Manager of the TDS to view Alice's pictures. Since David has had no previous interactions with Alice, the trusted distance from Alice to David is $d_{trust}(A, D) = d_{hop}(A, D) = 2$, and the Access Request Manager issues David an RFA certificate. David obtains authorizations from both Bob and Carol since they both know him. David then submits the authorizations to the Access Request Manager which verifies that he has satisfied the attestation requirements. David has now gained access to Alice's photo album. At this point, the affine distance from Alice to David is $d_{aff}(A, D) = -0.599$, which gives $d_{trust}(A, D) = 1.401$. This value captures the trust gained by David with respect to Alice through his attestations. Oscar, a friend of David, also wishes to view the photo album. Oscar's access request is denied because he has no previous interactions with Alice and so $d_{trust}(A, O) = d_{hop}(A, O) = 3$.

Suppose now that David chooses to republish Alice's photographs (to which he just gained access) as alb_{copy}. David attempts to publish the same photographs using

an accept limit of $c_{acc}(D, alb_{copy}) = 1$ and a reject limit of $c_{rej}(D, alb_{copy}) = 3$. The Data Leak Manager of the TDS detects that David is attempting to republish Alice's photo album under weaker confidentiality limits. To preserve the original limits set by Alice, the Data Leak Manager modifies the confidentiality limits submitted by David. Since the original data had a privacy setting of *strict*, the modified values are $c_{acc}(D, alb_{copy}) = 0$ for the accept limit and $c_{rej}(D, alb_{copy}) = 0.5$ for the reject limit. Now, although David added Oscar as a friend on the social network, assume that he has denied Oscar access to a few of his data objects, such that $d_{aff}(D, O) = 0.6$, resulting in $d_{trust}(D, O) = 1.6$. Oscar will still be unable to view the photographs, thus preserving Alice's original confidentiality limits. Note also that because David was originally attested to view the photographs, anybody wishing to view his copy of Alice's photographs must also be attested.

Chapter 5

Analysis

In this chapter, the complexity and security properties of PDAC are discussed. We shall see that the user and processing complexity of PDAC is comparable to that of other access control schemes and that the processing complexity for a data request grows linearly with network size.

5.1 User Complexity

A scheme for protecting personal data naturally solicits input from the user to set the parameters of the access control process. While user control is important, it is necessary to keep the required user effort low. One measure of user effort is the minimum number of user actions required to setup the access control policies. We compare PDAC with three alternatives: a simple public/private scheme, a hop based scheme, and a list based scheme. In a public/private scheme, a data object is tagged as *private* or *public*. Private data objects are restricted to the owner or a small set of users selected by the owner whereas public data objects are available to all users. As

discussed in Chapter 3, the hop-based scheme provides finer control over the access control process than the public/private scheme and exploits the structure of the social graph. In the list-based scheme, blacklists and whitelists associated with each object enumerate which users who should be denied or given access to it, respectively.

Let m be the total number of data objects and n be the total number of users in the system. In the public/private scheme, the user has to tag each object as public or private. Therefore, a public/private scheme requires at least m user actions. In a hop-based scheme, for each object the owner specifies a hop number that defines the sharing perimeter. Consequently, the hop-based scheme also requires at least m user actions. In the list-based scheme, default rules can be used to reduce the number of user specifications. For instance, a whitelist only scheme would imply "all other users" not enumerated in the list are denied access. Therefore, user i requires σ_i actions to set the parameters of the whitelist (or blacklist), where σ_i is number of users from the global population that user i cares to enumerate in his or her list (e.g., the size of the social neighbourhood). Suppose μ is the average size of the social neighbourhood over the full system, then the average number of user actions required by a list-based scheme is $m\mu$.

PDAC requires the owner of a data object to set two thresholds for each data object, which requires a total of $2m$ user actions. In addition, PDAC employs a user-specific pseudo-blacklist functionality by setting the per-friend distance to non-zero values. Suppose user i marks b_i other users as "bad" (i.e., wants to avoid sharing data with those users). The total number of user actions required for marking bad users is nb_n, where b_n is the average number of bad users designated by non-malicious users. So, the total number of user actions required by PDAC is $2m + nb_n$. Assuming

a relatively low number of malicious users in the social network on average, the $2m$ term dominates. Hence PDAC's user complexity is comparable to the alternatives.

5.2 Processing Complexity

To analyze the processing complexity, we must look at two cases: the complexity for a requester and the complexity for a publisher. First we analyze the processing complexity for a requesting user. We model the social network as an unweighted graph $G = (V, E)$, where V and E are the sets of vertices and edges in the graph, respectively. A requester y will keep a counter of the number of accepted and rejected access requests for each publisher x that y has sent an access request to. In order to compute the social neighbourhood success rate of y with respect to x, we consider only those interactions that y has with users from within x's social neighbourhood. The maximum number of publishers to whom y may send an access request, as well as the maximum number of users in x's social neighbourhood are both $|V|$. Therefore, since retrieving y's success rate information for each publisher takes $\mathcal{O}(1)$ time, computing $s_y(x)$ takes $\mathcal{O}(|V|)$ in the worst case. Computing y's individual success rate for x's data objects will be $\mathcal{O}(1)$. Therefore, the computation of $d_{aff}(x, y)$ has a processing complexity of $\mathcal{O}(|V|)$. A breadth-first search on G can compute $d_{hop}(x, y)$ in $\mathcal{O}(|E| + |V|)$. A requester falling in the Attestation Zone would need to contact at worst all n attesters in order to obtain at least k authorizations, giving a worst case complexity of $\mathcal{O}(n)$ to obtain the authorizations and $\mathcal{O}(k)$ to verify them. Since $k \leq n < |V|$, and computing the friend distances only takes $\mathcal{O}(1)$, the worst case processing complexity required for a user requesting data is $\mathcal{O}(|E| + |V|)$. Therefore, the processing complexity for a requester grows linearly with network size.

We now analyze the processing complexity for a user P publishing data obj. Because the majority of the computations involved when a user publishes data is performed during the similarity detection, the complexity is completely dictated by the type of document that the user is attempting to publish and the complexity of the comparison algorithm used. It is therefore difficult to determine a precise complexity analysis of this operation. However, simulations have shown [20, 23] that similarity detection algorithms can typically be computationally expensive (along the order of $\mathcal{O}(d^2)$ where d is the number of data objects) when performed on large document collections due to the one-to-all comparisons. To avoid this problem, PDAC performs the similarity analysis on $\mathcal{D}_P$, a subset of all data objects published on the TDS, which keeps the processing complexity low. The set $\mathcal{D}_P$ contains objects recently accessed by user P and is constructed using the successful requests stored in P's activity logs. The size of the current activity window can be changed so as to keep the average number of log entries fairly low, ensuring a small value for $|\mathcal{D}_P|$. The processing complexity can be further reduced by considering only those data objects that have the same type as obj, e.g., only considering JPEG files.

5.3 Security Analysis

Since the main goal of PDAC is to provide an access control mechanism for personal data, it is important that such a scheme be resistant to malicious activities and attacks. We now consider malicious attacks that can target PDAC. In particular, we discuss the scenario of a group of users colluding to obtain access to data which they should otherwise not be able to access. In this analysis we consider two types of attacks. The first type of attack focuses on manipulating the trust computation. The second type

of attack involves malicious attesters. We shall see that PDAC effectively prevents both types of attacks.

5.3.1 Manipulating the Trust Computation

Suppose that Alice is the owner of a data object *obj* to which Oscar, a malicious user, seeks access. Furthermore, assume that Alice's object is inaccessible to Oscar, indicating that Oscar either falls in the Rejection Zone or the Attestation Zone. If Oscar falls into the Attestation Zone, we assume that his request would eventually be rejected (i.e., Oscar is not able to acquire enough attester authorizations). Therefore, for simplicity we assume an Attestation Zone of size zero so that Oscar falls in the Rejection Zone. Attacks on a non-zero size Attestation Zone will be dealt with in Section 5.3.2. Alice sets the confidentiality limits for the object such that $0 < c_{acc}(A, obj) = c_{rej}(A, obj)$. The only way for Oscar to gain access to Alice's data object is to move to the Acceptance Zone from the Rejection Zone for that object. Oscar therefore needs to satisfy the following inequality in order to gain access:

$$d_{hop}(A, O) + d_{aff}(A, O) + d_{afdst}(A) + d_{pfdst}(A, O) \leq c_{acc}(A, obj) \tag{5.1}$$

To further simplify the analysis, we assume the friend distances $d_{afdst}(A)$ and $d_{pfdst}(A, O)$ are both zero. Using Equation 4.2, we get:

$$d_{hop}(A, O) + \lambda s_O(A) + (1 - \lambda)(\frac{r_{OA} - a_{OA}}{q_{OA} + \Delta}) \leq c_{acc}(A, obj) \tag{5.2}$$

In Inequality 5.2, the only parameter which Oscar can actually manipulate without Alice's knowledge is the social neighbourhood success rate $s_O(A)$ because it is based

on his request history to Alice's social neighbourhood. In contrast, the third term on the left side is based on the direct interaction between Oscar and Alice. Therefore, Oscar could potentially collude with users in Alice's social neighbourhood in order to improve his social neighbourhood success rate and gain access to the data object. Let us re-examine Equation 4.1 but substitute δ for the raw success rate (assume $q_y \neq 0$):

$$s_O(A) = \delta(\frac{1}{1 + e^{\beta - p/\alpha}}) \tag{5.3}$$

In order for Oscar to enter the Acceptance Zone and gain access, he needs a negative $s_O(A)$ value so that $d_{trust}(A, O)$ is reduced. A non-negative $s_O(A)$ value does not help Oscar because this will only keep him in the Rejection Zone. Therefore, Oscar can only gain access if $s_O(A) \in (-1, 0)$. We capture this in Inequality 5.4:

$$\delta(\frac{1}{1 + e^{\beta - p/\alpha}}) \leq \Gamma, \text{ for some } \Gamma \in (-1, 0) \tag{5.4}$$

Since the scaling factor of $s_O(A)$ depends on p, the number of unique data posters in the access history, Oscar not only needs to find users willing to collude with him to get a δ value closer to -1, but also needs enough colluders so that the scaling factor is maximized. The graph displayed in Figure 5.1 plots the number of colluders required in order to gain access to the data for a given Γ value. The plot assumes a best case scenario (for Oscar) of $\delta = -1$. For illustrative purposes, the system parameters were set to $\alpha = \beta = 5$.

From figure 5.1 we note that even for small Γ values, a malicious user would need to collude with a considerable number of users. For example, if $\Gamma = -0.05$ then Oscar would require at least 11 users with whom to collude in order to access Alice's

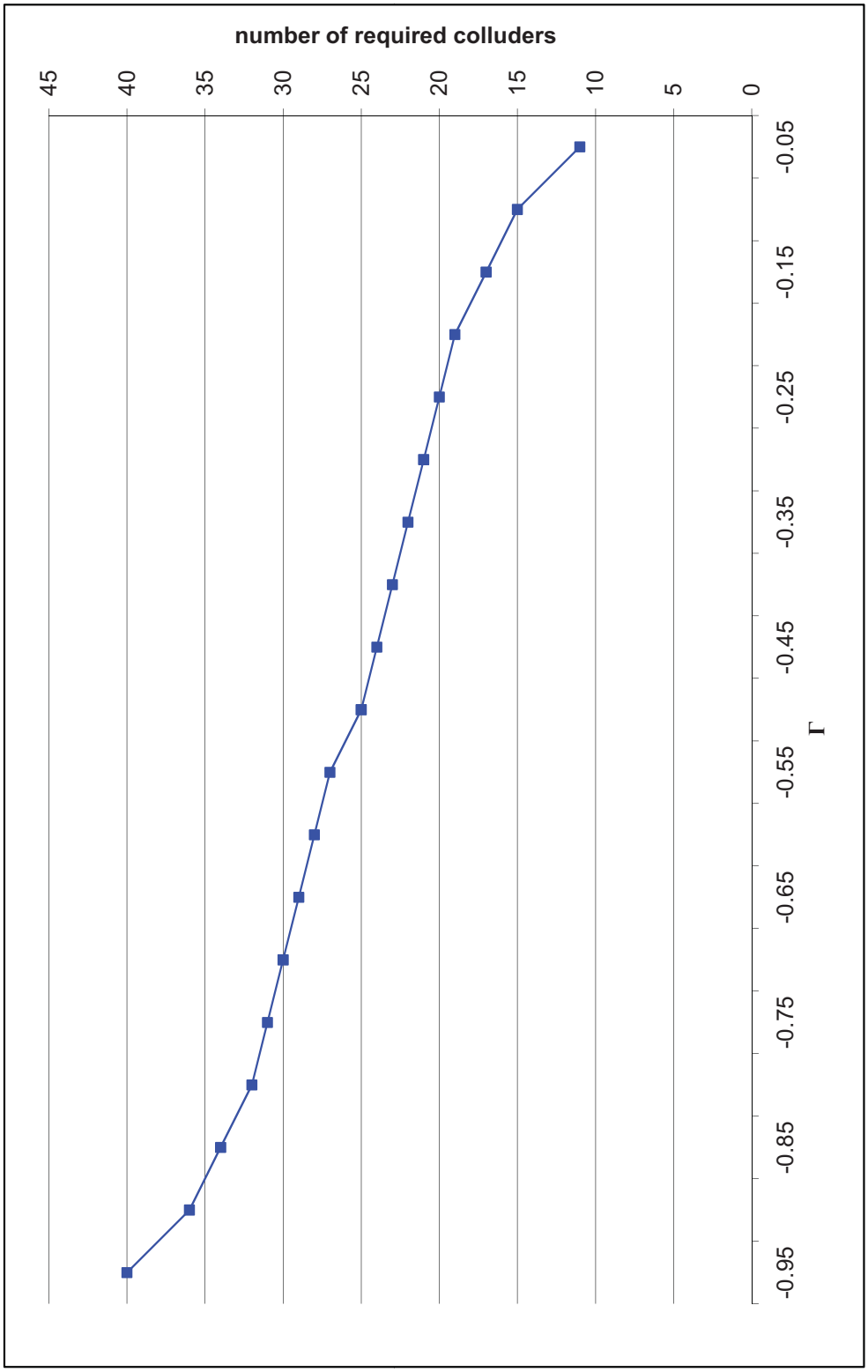

Figure 5.1 The minimum number of required colluders for a given Γ value. This graph shows that as Γ approaches -1, the minimum number of required colluders increases.

data object. Furthermore, the users with whom Oscar needs to collude must be from Alice's social neighbourhood. This makes Oscar's task significantly harder because he would be less likely to find the appropriate number of colluders among users who are socially close to Alice and each other. Finally, note that as Γ gets closer to -1, it becomes more difficult for an attacker to gain access. The reason for this may be found by rearranging Inequality 5.2 and substituting Equation 5.3 to derive a value for Γ:

$$\delta(\frac{1}{1+e^{\beta-p/\alpha}}) \leq \frac{c_{acc}(A,obj) - d_{hop}(A,O) - (1-\lambda)(\frac{r_{OA}-a_{OA}}{q_{OA}+\Delta})}{\lambda} = \Gamma \qquad (5.5)$$

Thus, we can see that Γ will be closer to -1 with lower accept limits, larger values for hop distance, and larger positive individual success rates. This makes it more difficult for untrustworthy users to gain access. Intuitively, this makes sense since lower accept limits indicate that Alice considers the data object to be highly sensitive and thus only wants highly trusted peers to gain access. A large hop distance indicates that Oscar is far removed from Alice's social neighbourhood. Finally, and perhaps most importantly, a large positive individual success rate indicates that Alice has rejected most of Oscar's access requests, indicating Alice's distrust of Oscar.

5.3.2 Malicious Attesters

In the second type of attack, Oscar attempts to collude with the attesters for data object *obj* in order to gain access. Assume that Oscar falls into the attest zone and that malicious users exist among the attesters. If Oscar is able to collude with enough malicious attesters to obtain the minimum number of authorizations required, then Oscar can gain access. A solution to prevent such an attack is to require authorization

from a majority of the attesters before granting access. The attester majority solution is effective especially in the context of social networks because data owners will typically choose highly trusted family members or friends and it is therefore unlikely that the majority of the attesters will be malicious. Users must therefore choose attesters for their data objects wisely, ensuring that the attesters chosen for a particular data object are at least as trustworthy as the confidentiality of that data object.

Chapter 6

PDAC Implementation

In this chapter we discuss the design of the working prototype of the PDAC scheme that has been implemented using the Ruby programming language. PDAC is implemented built on top of a social network emulator called the *Social Utility Networking Environment* (SUNE). SUNE has a hybrid architecture that combines client/server and P2P paradigms. This is ideal for PDAC due to the shared responsibilities of the TDS components and attesters in determining whether a requester should be given access to a particular data item.

6.1 System Architecture

The hybrid architecture of SUNE means that functionality is evenly distributed among the server and the peers in the social network. PDAC is implemented as an application on top of SUNE that uses the underlying framework to determine trusted distance. The TDS is implemented as a service that is constantly running on a central SUNE server. The distributed architecture discussed in Chapter 4 is not implemented in

this prototype. The WEBrick Ruby library was used to build the SUNE server. User functionality in PDAC is implemented on the peer side as a plugin (not unlike the application concept in `facebook.com`) that communicates with the TDS service on the server.

6.2 Peer Functionality

The peer functionality in PDAC involves setting the security settings of a data to be published, sending access requests to the server, obtaining authorizations from attesters, processing signature requests from other users, and keeping access request logs. The main objects involved are the Data Store Client object, the Attestation Manager object, the Access History Manager object, the Blacklist Manager object, the Friend Distance Manager object, and the Notification Manager object.

6.2.1 Data Store Client

The Data Store Client is the main peer object that communicates with the TDS on the server. All communications between a peer and the server occur through the Data Store Client. It is responsible for uploading data objects to the TDS to be published as well as sending access requests to other peer's data objects. When publishing a data object, the user is required to define the accept and reject limits, choose attesters, and specify the number of attester authorizations required to obtain access. The user may also select the dissemination setting, although the system assigns a *strict* dissemination setting by default. The user has the option of requiring manual attestation for an uploaded data object. A manually attested data object requires its attesters to manually approve a peer that has requested data access. During the

attestation process (manual or otherwise), the Data Store Client is responsible for sending the attester authorizations to the TDS. In conjunction with the Notification Manager, the Data Store Client is also responsible for notifying the user if they have been selected as a manual attester for an object, if a manual attester authorization request is pending, and whether their access request has been approved or rejected.

6.2.2 Attestation Manager

The Attestation Manager is responsible for all peer-side attestation functionality. This object maintains a list of data objects for which this peer is an attester. It also stores the attester criteria for that data object which is used to determine whether to approve an attester authorization request. If the peer that initiated the attester authorization request satisfy the criteria, the Attestation Manager automatically signs the authorization request, otherwise the request is ignored. Manually attested data objects require the user to inform the Attestation Manager on whether to sign the authorization request. For a requester, the Attestation Manager is responsible for contacting the attesters for the desired data object and collecting the necessary amount of attester authorizations. Once the required number of authorizations have been obtained, the Attestation Manager gives the set of attester authorizations to the Data Store Client to send to the TDS.

6.2.3 Access History Manager

The Access History Manager is responsible for storing the activity request logs for a peer. It updates the activity log with every data access request sent to the TDS. The Access History Manager also computes the affine distance between users using

the information stored in the activity logs. The affine distance is computed using a user's request history with the publisher's social neighbourhood. A 2-hop social neighbourhood is used in this implementation. The Access History Manager removes entries from the activity logs once they fall outside of the current activity window. This implementation uses a window size of one week.

6.2.4 Blacklist Manager

The Blacklist Manager allows the user to immediately prevent certain peers from gaining access to his or her data items regardless of their trusted distance. This implements the pseudo-blacklists in PDAC by assigning a large value to the user's per-friend distance for the peer to be blacklisted. Peers on a user's blacklist are barred from accessing the user's data objects. In addition, any attester authorization requests from peers who are on the attester's blacklist are automatically rejected. A user's blacklist is shared among his or her 1-hop friends. This allows the user and his or her immediate friends to quickly discover which users in the social neighbourhood have been acting maliciously. However, blacklist sharing can result in the rest of the social neighbourhood imposing their blacklists on each user. To combat this, this implementation allows each user to override the peer defined blacklist with a local user defined one.

6.2.5 Friend Distance Manager and Notification Manager

The Friend Distance Manager is responsible for storing the friend distances. These values are sent to the TDS to be used in computing the trusted distance. The user has the option of setting the all-friend distance, which applies to all users attempting to

access his or her data, as well as the per-friend distance, which applies to a particular peer. Since per-friend distances are set to zero by default, only non-zero per-friend distances are actually stored. The Notification Manager is responsible for displaying messages to the user, informing them of access request decisions, attestation assignments, and pending attestation requests.

6.3 TDS Server Functionality

TDS server functionality in the PDAC prototype include storing published data items, computing trusted distance between a requester and a publisher, verifying attester authorizations, and detecting data leaks. The main objects involved are the Content Manager, Access Request Manager, Trust Manager, Peer Manager, Topology Manager, and Data Leak Manager.

6.3.1 Content Manager

The Content Manager is responsible for storing all the data objects on the TDS. Along with the data object itself, the Content Manager stores the metadata associated with that data object. This metadata includes the owner ID of the data object, the accept and reject limits, attestation type (manual or automatic), the list of attesters along with the number of required attester authorizations, and the dissemination setting. The Content Manager can be easily configured to use an open source database such as PostgreSQL.

6.3.2 Access Request Manager

The Access Request Manager is responsible for processing all access requests that are sent to the TDS. It calls functions of the Trust Manager to compute the trusted distance between the requester and the publisher. Using the computed trusted distance, the Access Request Manager determines which protection zone the requester falls under with respect to the publisher of the data object. It issues the RFA certificate to requesters that fall in the Attestation Zone, and confirms that the RFA certificate returned by the requester has been signed by enough of the assigned attesters for that object. If the Access Request Manager has determined, based on the data object's security settings, that the requester should be granted access to the data object, it will retrieve a copy of the data object from the Content Manager and provide the requester with a copy.

6.3.3 Trust Manager, Peer Manager, and Topology Manager

The Trust Manager is responsible for computing the trusted distance between a publisher and a requester. The Access History Manager of the requester and Friend Distance Manager of the publisher are both contacted in order to retrieve the affine and friend distances, respectively. The TDS uses the Peer Manager to contact all the users in the network. The Topology Manager maintains the overall topology of the network. The Trust Manager contacts the Topology Manager to obtain the hop distance between the publisher and the requester. The Trust Manager then computes the total trusted distance between the publisher and the requester.

6.3.4 Data Leak Manager

The Data Leak Manager is responsible for detecting and preventing data leaks. To implement the data leak detection functionality of the Data Leak Manager, an open source application called DuMP3 [29] was used. DuMP3 uses a variety of different fingerprinting algorithms to determine file similarity. The algorithms used are capable of detecting similar text files, images, audio, and video files. Because a different fingerprinting algorithm is used for each type of file, the similarity thresholds are different for each filetype. This also means that files are only compared to other files with similar file types. For example, image files are only compared to other image files. This reduces the number of comparisons required when attempting to detect a data leak. When a user submits a data object to be published on the TDS, the Data Leak Manager compares it to the data objects that user has accessed recently. If a data object is found with a high enough similarity score to the data object to be published, the Data Leak Manager adjusts the accept and reject limits of the according to the dissemination setting selected by the publisher of the original data object. The Data Leak Manager then sends the data object to the Content Manager for storage.

Chapter 7

Simulation Results

In this chapter, the results of simulations of the PDAC scheme are presented and discussed. The simulations were performed to evaluate the effectiveness of PDAC in enforcing data access privileges.

7.1 Simulation Setup

In this section we see the results from the simulations that were performed to evaluate PDAC and compare it with the simple protection scheme discussed in Chapter 3. Recall that in a hop-based access control scheme, the access decision is based solely on hop distance, so that a requester is given access if his distance from the owner is within the hop limit set by the owner. To setup the simulation, portions of a social graph obtained from `myspace.com` were used (the super-user Tom was removed from the data). A request generator (an oracle) was then used to generate a set of requests along with the expected access outcomes. The set of requests generated by the oracle was presented to both the PDAC and hop-based methods, and the access

decision of both methods were compared to the oracle's decision for evaluation. Three metrics were defined for evaluation: success rate, false positives, and false negatives. The success rate is defined as the fraction of access decisions that match the oracle's decision. False positives are defined as the fraction of requests that were given access by each scheme but were denied access by the oracle. False negatives are defined as the fraction of requests that were denied access by each scheme but given access by the oracle. Malicious users were introduced into the network to evaluate the effectiveness of PDAC in detecting and denying access to these malicious users. A small, randomly selected subset of the users in the network were marked as malicious users. The success rate of malicious users is defined as the fraction of malicious requests that gained access.

The requests and outcomes are generated by the oracle based on two types of distributions: uniform or hop-based discrete. For the hop-based discrete distribution we define three distinct probability density functions. In each distribution, we assign a probability to a hop value $h > 0$. The probabilities for each distribution are shown in Table 7.1. The "steep" and "shallow" tags are to indicate the slope of the probabilities with respect to hop distance. Discrete steep has high probabilities for closer hop distances, and as the distributions get shallower, the probability distributions get closer to uniform. To generate a request, a publishing user P is first selected at random. The requester R is then chosen according to either the discrete shallow, discrete shallower, or uniform distribution. For the uniform distribution, requesters were selected without regard to hop distance. For the hop-based discrete distributions, the probability that P receives an access request from user R who is h hops away from P is as given in Table 7.1. The outcome of each request is similarly determined by

using either the discrete steep or discrete shallow distribution (i.e., the probability that R's request to access P's data objects is granted, where R is h hops away from P, is as given in Table 7.1). All requests made by a user that has been marked as malicious were rejected by the oracle regardless of hop distance. For each of the cases, we denote the distributions used as (request distribution, outcome distribution).

	h					
Distribution	1	2	3	4	5	≥ 6
Discrete steep	0.5	0.35	0.08	0.04	0.02	0.01
Discrete shallow	0.3	0.3	0.3	0.05	0.03	0.02
Discrete shallower	0.25	0.25	0.25	0.15	0.07	0.03

Table 7.1 Hop-based discrete probability distributions used to determine requests and outcomes in the simulation.

For the purposes of the simulation, the parameters of both PDAC and the hop-based scheme were fixed for each outcome distribution. For PDAC, the control parameters for $s_y(x)$ were set to $\alpha = \beta = 5$. The all-friends distance in PDAC were set to 0 for the duration of the simulations. Per-friend distances were shared among the 1-hop friends of each user. Each simulation run consisted of 50000 requests and each simulation began with a warmup period where the oracle reveals the outcome of each request to both methods. This was done so that both methods could self-calibrate.

7.2 Simulation Results

Table 7.2 shows the results for different simulation runs using a social network with no malicious users. The results show that when no malicious users exist in the network,

most of the runs exhibited lower success rates for PDAC than the hop-based scheme with lower false positives and higher false negatives. PDAC had a better success rate than the hop-based scheme when the outcome distribution was discrete shallow. For that distribution, the hop-based scheme had a very high incidences of false positives, and very low incidences of false negatives.

		PDAC			Hop-Based		
Request Dist.	Outcome Dist.	Success Rate	False Positive	False Negative	Success Rate	False Positive	False Negative
discrete shallower	discrete steep	0.684	0.131	0.185	0.751	0.176	0.074
discrete shallow	discrete steep	0.645	0.149	0.206	0.697	0.230	0.073
discrete shallow	discrete shallow	0.671	0.191	0.139	0.593	0.400	0.008
uniform	discrete steep	0.926	0.025	0.049	0.946	0.004	0.050
uniform	discrete shallow	0.887	0.024	0.090	0.863	0.088	0.049

Table 7.2 PDAC vs. Hop-Based Scheme Simulations Results with No Malicious Users.

Table 7.3 shows the results using a social network that contains 10% malicious users. In almost all of the simulations, the PDAC had a higher success rate with lower false positives and higher false negatives. PDAC reduced the incidence of false positives by at least half in most cases. For the (uniform, discrete steep) case the hop-based method has better values for almost all categories. In almost all the cases, the hop-based scheme has lower success rates than the simulations which contained no malicious users. In contrast, PDAC resulted in higher success rates for all runs when there are malicious users than when there are no malicious users. False positives

for PDAC also dropped across the board from the false positives obtained with no malicious users in the network. However, the hop-based scheme resulted in higher false positives when malicious users were introduced into the network. For the malicious success rates, PDAC performs much better than the hop-based method for all cases except for the (uniform, discrete steep) case.

Table 7.4 compares the malicious success rates of PDAC when malicious users have different initial levels of notoriety. Initial notoriety in this context means how well known the malicious users are in the network at the start of the simulation. An initial notoriety of $x\%$ indicates that $x\%$ of the users in the network know from the start which users are malicious. The results show that even with only 10% initial notoriety, the malicious success rate is still under 7%.

7.3 Discussion of Results

When there are no malicious users in the network, PDAC appears to perform worse than the hop-based scheme due to the lower success rates. Despite this, PDAC reduces the incidence of unauthorized data access (as evidenced by the lower number of false positives) when compared to the hop-based scheme. PDAC's higher number of false negatives indicates that it is more conservative than the hop-based scheme, seemingly erring more on the side of caution when deciding whether or not to grant access. The hop-based scheme performs worse when the outcome distribution is discrete shallow because the oracle has a lower probability of accepting requests from users with a lower hop count. The probability that the oracle would grant a request from a user that is closer to the publisher for a discrete steep distribution is higher than for the discrete shallow distribution. Since the hop-based scheme only bases its decision on

		PDAC				Hop-based			
Request dist.	Outcome dist.	Success rate	False positive	False negative	Malicious success rate	Success rate	False positive	False negative	Malicious success rate
discrete shallower	discrete steep	0.761	0.111	0.128	0.010	0.697	0.241	0.062	0.615
discrete shallow	discrete steep	0.687	0.132	0.181	0.034	0.669	0.270	0.062	0.539
discrete shallow	discrete shallow	0.690	0.183	0.127	0.064	0.529	0.463	0.008	0.914
uniform	discrete steep	0.934	0.022	0.044	0.001	0.950	0.003	0.047	0.005
uniform	discrete shallow	0.894	0.025	0.082	0.025	0.863	0.093	0.044	0.144

Table 7.3 PDAC vs. Hop-Based Scheme Simulations Results with 10% Malicious Users.

		Malicious success rate	
Request dist.	Outcome dist.	50% Initial notoriety	10% Initial notoriety
discrete shallow	discrete steep	0.007	0.034
discrete shallow	discrete shallow	0.022	0.064
random	discrete shallow	0.008	0.025

Table 7.4 Comparison of malicious success rates for 50% initial notoriety and 10% initial notoriety.

hop distance, and only grants access to those users within the hop threshold, the hop-based scheme causes more false positives to occur, lowering the success rate.

With 10% malicious users in the network, we see that for almost all of the runs, PDAC performs much better than the hop-based scheme. PDAC appears to perform better when there are malicious users in the network than when there none. The lower malicious success rates for PDAC indicate that it is much better suited to dealing with malicious users than the hop-based scheme. This is probably due to PDAC's use of interaction history between the users. Once again, the lower number of false positives indicates that PDAC reduces the incidences of unauthorized data access when compared to the hop-based scheme. Like the runs with no malicious users, the higher incidence of false negatives in PDAC means that it is more conservative than the hop-based scheme, indicating a more protective mechanism. The only case that favours the hop-based scheme is the (uniform, discrete steep) case. A possible explanation is because the users who are far away from the source are equally likely to make a request as those users who are within the source's neighbourhood. Given

the discrete steep outcome distribution, the oracle will reject requesters that are not close to the source. However since the hop-based scheme also rejects requesters that are not within the hop threshold (which are usually set to values such that users in the source's neighbourhood are given access), for this case the decisions made by the hop-based scheme will closely match that of the oracle's. Despite this, the PDAC still performs quite well, giving a success rate of over 92%.

As for initial levels of notoriety, we see that PDAC reduces the malicious success rates as malicious users become more well known. This shows that PDAC's pseudo-blacklist sharing functionality is effective in spreading news about malicious users in the network. Even with a low level of initial notoriety, blacklist sharing keeps the malicious success rate low because users eventually learn who the malicious users are even if they have not directly interacted with them.

Chapter 8

Related Work

In this chapter, we discuss other trust-based access control systems and compare it to PDAC. One system in particular, the Rule-Based system developed by Carminati *et al.* [5], is discussed in greater detail since, like PDAC, it is designed specifically for use in a social network context.

8.1 Rule-Based Access Control

In [5], Carminati *et al.* present an access control model for web-based social networks, where policies are expressed as constraints on the type, depth, and trust level of existing relationships. The model uses a graph representation of the social network with users as nodes to express the different policies. For the relation-type the authors propose to extend the graph representation to support relationship types. The depth of a relationship between two users corresponds to the shortest path among all possible paths with in/direct relations of the given type. The trust level between two users a and b for a given relationship type is defined as how trustworthy user a considers user

b with respect to the given relationship. The details of the formulas for calculating the trust level are not considered by the authors. Finally, a relationship is represented by a tuple, where its components denote the users participating in it, along with the type, depth, and trust level of the relationship itself.

In their proposed model, each owner defines the access rules according to his or her preferences, while each requester is in charge of verifying to the owner whether he or she is authorized to access a given object. When a node requests access to a resource, the owner releases the access rules to the requesting node. In order to access the resource, the requesting node must provide the owner with a proof showing that it actually satisfies the policies. Proofs are generated using a reasoner [30], which allows the owner to locally verify whether the resource should be released or not. To prevent the requesting node from maliciously forging the proof, the authors propose the use of certificates. Whenever two users establish a new relation they create and sign a certificate stating there exists a direct relation with a certain trust level. Because generating proofs can become incredibly complex, especially when proving assertions regarding trust, special *Central Nodes* are introduced to manage the certificates and to provide reasoner functionality for the user generated requests.

One similarity between the Rule-Based access control model and PDAC is that both schemes exploit the structural properties of the social network in the access control model. In PDAC, the social-graph-based hop distance between the users, in addition to the activities occurring in social neighbourhoods, are utilized in mapping the users into zones, which determine whether a user should be given access. In the rule-based scheme, the type and depth of relationships in the social network are used to express the access control policies. However, there are important differences

between the two schemes. The zone mapping in PDAC creates a dynamic overlay of relationships on top of the static social network. This overlay model allows for dynamic relationships which adapts according to previous user actions. The rule-based scheme relies on a static social network, whereas in PDAC decisions regarding the resource access can directly affect the trust level stored in the relationship overlay. Thus it can be argued that PDAC correlates more closely to real life situations. The dynamic nature of relationships and how they affect resource access is better captured by the adaptive overlay in PDAC than by the static nature of the rule-based scheme. Another important difference between PDAC and the rule-based scheme is that PDAC considers data leaks. The Data Leak Manager enforces the access control policy set by the original owner of the resource. This ensures that users who have gained access to the resource cannot use the social network to redistribute the protected data. The rule-based scheme does not consider this issue.

8.2 Adaptive Trust Negotiation

In [31], Ryutov *et al.* propose an Adaptive Trust Negotiation and Access Control (ATNAC) framework to protect sensitive resources in electronic commerce. It relies upon two components: The Generic Authorization and Access-control API (GAA-API) [32] and TrustBuilder [33]. The GAA-API provides adaptive access control that captures dynamically changing system security requirements. The TrustBuilder system automates trust negotiation between strangers on the Internet.

ATNAC uses the GAA-API to verify a requester's credentials to determine whether to grant or deny the request. If the required credentials are unavailable, or a significant system threat exists, the API computes a suspicion level for the requester and

forwards it, along with a set of required credentials, to the TrustBuilder system. The requester is notified that further proof is required and initiates trust negotiations with the TrustBuilder server. The TrustBuilder server uses the suspicion level to determine the credential release policies. Once the client's credentials are verified and the TrustBuilder server's policies are satisfied, the TrustBuilder server sends a recommendation to the GAA-API, which then makes the final decision based on the trust negotiation results.

The use of third-party trust credentials to obtain access is similar to the attestation process of PDAC. Both the trust negotiation protocol of TrustBuilder and the attestation process of PDAC allow a service provider or resource owner to gain information about an unknown requester. However, because ATNAC is meant for use on the open Internet where the relationship between a requester and a resource owner are often unknown, third-party credentials are required to determine the trustworthiness of a requester. The social network domain of PDAC allows it to exploit the trust inherent in social relationships to determine whether a requester should be granted access.

8.3 TrustBAC

In [34], Chakraborty and Ray present a trust-based extension of RBAC for open systems called *TrustBAC*. TrustBAC assigns trust levels to users based on three factors – experience, knowledge, and recommendation. The experience component is based on the user's past behaviour. The knowledge component is concerned with knowledge about other characteristics of the user, such as credentials, for example. It has two parts – direct knowledge and indirect knowledge. The weighted sum of direct and

indirect knowledge constitutes the knowledge component. The recommendation component is evaluated based on recommendation values provided by others about a user, and the recommendation values are weighted by the reputation of the recommender. Using these parameters, the system computes overall trust levels, and assigns them to roles. Roles, in turn, are assigned to permissions, just as in traditional RBAC models.

TrustBAC is very similar to a trust-based scheme developed by Almenárez *et al.* for pervasive devices called TrustAC [35]. A Pervasive Trust Management (PTM) model is used to assign trust degrees to users. This scheme is implemented as an XACML Profile [36] by using Sun's XACML Implementation [37]. The XACML standard facilitates inter-operability between authorization systems, and allows for a distributed scheme.

PDAC is somewhat similar to the TrustBAC and TrustAC schemes in that trust levels are linked to permissions. The zone concept of PDAC determines the permissions that a requester has for a protected resource. However, the link is more direct in PDAC, since there is no concept of roles. Also, both TrustBAC and TrustAC assign a universal trust value for each user, which indicates how much the system as a whole trusts the requester. PDAC assigns trust to relationships between users, which indicates how each individual user trusts the requester. This provides more flexibility and is more realistic, since a particular user may be considered untrustworthy by one group of users, but be highly trusted by another.

8.4 Trusted Computing

In [38], Sandhu and Zhang discuss a peer-to-peer access control architecture that uses Trusted Computing technology. Their system considers the integrity and trust of

platforms and applications rather than user property based policies. The proposed architecture uses trusted hardware on both the client and server side to ensure security of resources, and allow for platform-to-platform propagation of trust. A Trusted Platform Module (TPM) chip is used to perform cryptographic functions such as random number generation and RSA key generation. The security key used to perform these functions are stored in the TPM and is never released. Each platform also has a Trusted Reference Monitor (TRM) which seals secrets and policies using the functions provided by the TPM. Secure channels are then used to transfer secrets between trusted platforms.

While this scheme certainly provides a higher level of security than PDAC, it is impractical in a social networking context. The dependence of the scheme on Trusted Computing technology makes it very expensive to implement. Each user in the network would require a certified platform that uses special hardware in order to participate in a network employing this scheme. Also, because this scheme is platform dependent, the user is required to migrate access policies every time the user switches machines. This is very inconvenient for the user.

8.5 Other Trust-Based Access Control Mechanisms

In [39], Tran *et al.* present a trust-based extension of DAC for peer-to-peer file sharing. Two threshold values (one based on trust scores and another based on contribution scores) are associated with each file. A peer seeking access to the file should have trust and contribution values exceeding these threshold values. The trust parameter models the malicious behaviour, while the contribution value models the cooperative behaviour. The trust and contribution scores are computed using direct and indirect

measurement values. This helps to classify both known and unknown users. PDAC also classifies both known and unknown users by using both affine and hop distance to determine trustworthiness.

In [40], Adams and Davis discuss a trust based access control system for mobile ad-hoc collaborative environments. The system combines node reputations and environment risk to make access decisions. A node's reputation is the perception that peers form about a node, and environment risk is a measure of how risky an action is likely to be given the current state of trust events in the environment. A user's ability to access a resource can change as her reputation changes. Trustworthiness of a particular node is dependent upon that node's historical behaviour. This is similar to the affine distance concept of PDAC.

In [41], Dimmock proposes SECURE, a trust-based generic decision making framework for global computing. In SECURE, an access control manager grants or denies permission for principals to execute actions on the contacted host. Any decision in SECURE considers the trust in the requesting principal and the risk of granting such request. The risk is defined as the combination of the cost and likelihood of all possible outcomes for the requested action. Trust is computed using observations of previous interactions and recommendations of other users. This is similar to how PDAC uses interactions between the requester and the owner's social neighbourhood to determine the trustworthiness of the requester.

In [42], Bhatti *et al.* propose a trust-based context-aware access control model for web-services called X-GTRBAC. X-GTRBAC is an XML augmented extension of the Generalized Temporal Role Based Access Control (GTRBAC) model [43]. GTRBAC provides a generalized mechanism to express periodic and duration constraints

on roles, user-role assignments, and role-permission assignments. The XML extension allows policy enforcement in heterogenous and distributed environments. In X-GTRBAC, trust is interpreted as the level of confidence associated with a user based on certain certified attributes. The certification is provided by trusted third parties such as a PKI certification authority (CA) which is then used to assign roles to users. The system can then adjust the trust levels based on the user's activity profile. The use of a CA contrasts with the attestation process of PDAC, which uses relationships within the social neighbourhood to determine access rights.

8.6 Reputation-based and Policy-based Trust Management

In [44], Bonatti *et al.* propose using a combined approach of both Reputation-based and Policy-based trust management to enhance existing trust management tools. They argue that integrating the two together results in a versatile trust management language that is capable of addressing both structured organizational environments as well as unstructured user communities. Their proposed approach envisions a system that combines user credentials and rules with numerical trust estimates computed using a large number of sources. PDAC appears to be such a system. PDAC uses numerical trust estimates to determine whether a user should be given access to a particular resource. Ultimately, user credentials are only as trusted as the party that issues them. The RFA certificates used in PDAC act as user credentials for requesters, and the attesters that sign those certificates are presumably highly trustworthy, since they are personally selected by the resource owner.

Chapter 9

Conclusions and Future Work

In this chapter, we discuss some future work that can be performed to further extend and improve PDAC. Some concluding statements about PDAC follow.

9.1 Future Work

The PDAC implementation discussed in this thesis was written as a proof-of-concept prototype. As such, certain aspects of it can be further improved and more functionality added. We now discuss a few of these enhancements.

9.1.1 Mnemonic Labels

Throughout this thesis, we have assumed that trusted distance is measured as a real number greater than zero. For example, a *friend* is at trusted distance 1.0 and *friend-of-a-friend* is at 2.0. Similarly, we can represent a *good friend* by a number less than 1.0 (for example, 0.9), a *very good friend* by a number less than that used for a good friend (for example, 0.8), a *good friend-of-a-friend* by a number less than 2.0 (for

example, 1.8), and so on. To improve the usability of the system, we need mechanisms that will accept confidentiality specifications in fuzzy mnemonic labels such as *very good friend* or *good friend*. Such a mnemonic labelling scheme allows the publisher to more intuitively specify the confidentiality threshold values for a data object. For example, a publisher may choose to define the confidentiality threshold values for a data object as [*good friend*, *very good friend-of-a-friend*]. This specification allows the publisher to unconditionally provide access to someone who is at least a good friend and reject anyone who is not at least a very good friend-of-a-friend. Such mnemonic labels would make the system more user-friendly, making it more appealing to users. Further research would be required to determine the values that would be appropriate for each mnemonic label.

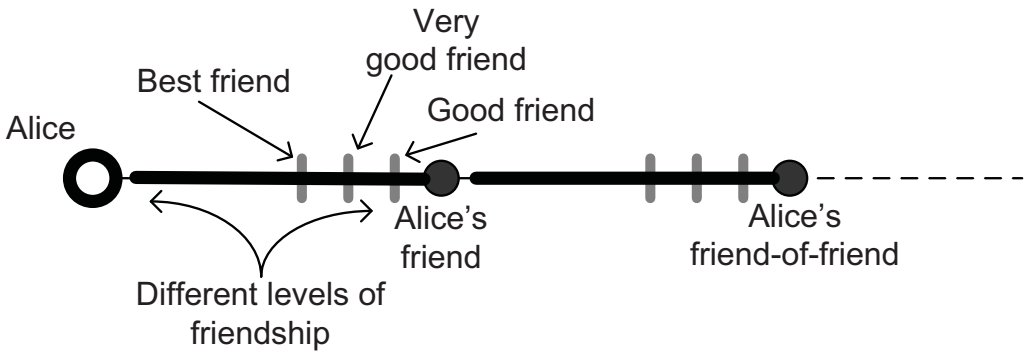

Figure 9.1 Illustration of trust distance labels.

9.1.2 Data Context Tagging

PDAC, as presented in this thesis, assumes a single context for data publishing. However, the scheme can be extended to handle multiple contexts. As discussed in Chapter 4, the trust measures are computed using data from prior user activities. The

entries in the activity log can be tagged to indicate the context under which they took place. Using the tagging information, past activities which are not relevant to the current context can be filtered out. Consider an example where Bob is seeking access to data objects posted by Alice that are categorized as `work`. If Bob is a relative of Alice, all prior interactions between Bob and Alice that are logged with a `family` tag do not affect the `work` trusted distance from Alice to Bob. It will be therefore be highly unlikely that Bob will be able to access any company sensitive information. Conversely, Carol, a coworker of Alice, will have all her prior interactions with Alice logged with a `work` tag, and so should be able to gain access to the work related data that Alice has posted. The interaction tags can also be used to deduce the nature of the relationship between the two users. Additionally, the context tag can be used to suggest a list of attesters which would be more appropriate given the content of the data object. For example, for a data object with a `work` tag, the system would suggest a list of coworkers as attesters. More analysis is necessary to determine how to incorporate this feature into PDAC.

9.1.3 Like-Minded Users

When computing the affine distance between a publisher and requester, the current PDAC scheme considers interactions between the requester and all users in the publisher's social neighbourhood. A better indicator of a requester's trustworthiness would be to consider interactions between the requester and like-minded users in the publisher's social neighbourhood. Like-minded users can mean users who have made similar access control decisions as the publisher, or it may mean other users who have similar interests. In addition to giving the publisher more information about

a particular requester, the publisher is assured that the resulting outcomes of those access requests would have been the likely outcome had the publisher made the decision himself or herself. Determining which users have similar interests or have made similar access control decisions and incorporating it into the PDAC scheme would be an interesting future project.

9.1.4 Enhanced Hop Distance

Hop distance, as defined in this thesis, is computed using the shortest number of hops between any two users on the social network. However, if the network contains a few very popular users of whom everyone is a direct friend, this can result in a hop distance of at most 2 between any two users in the social network. This thesis has assumed a social network without such super-users. A possible enhancement to PDAC in order to deal with super-users is to average the distances of all the paths between any two users in the social network. Future work could explore the effect of an enhanced hop distance upon PDAC's performance.

9.2 Conclusion

This thesis has sought to develop an effective access control scheme for use in social networks. The Personal Data Access Control scheme discussed herein has addressed certain issues with existing access control schemes. First and foremost, it allows a publisher to differentiate among his or her friends in the social network, so that the more trustworthy friends are allowed more access to his or her data than the less trusted friends. It does this by using previous interactions along with hop distance to determine which users are trustworthy enough to gain access to his or her data

objects. PDAC also allows users to exploit the information inherent in social networks by sharing this information among their peers. Finally, PDAC addressed the issue of data leaks, and introduced a mechanism to ensure that malicious users cannot use the social network itself to circumvent the initial security settings of a data object.

Simulations results show that PDAC performs better than a simple hop-based scheme in allowing legitimate friends access to a publisher's data while at the same time preventing malicious users from gaining access. Although PDAC was designed primarily for social networks, their increasing use as application platforms means that PDAC can be adapted for many other situations where data is required to be disseminated in a secure fashion. As social networks become more popular, they will become an increasingly important method of communication. Because of this, it is of vital importance that we start considering effective and flexible access control schemes to protect the data stored in social networks.

Appendix A

PDAC UML Diagrams

This appendix contains UML diagrams that describe the PDAC prototype. Class diagrams for the Server and the Peer are provided. The major scenarios in PDAC use – publishing a data item, requesting access to a data item, and obtaining attester authorization – are displayed using a sequence diagram.

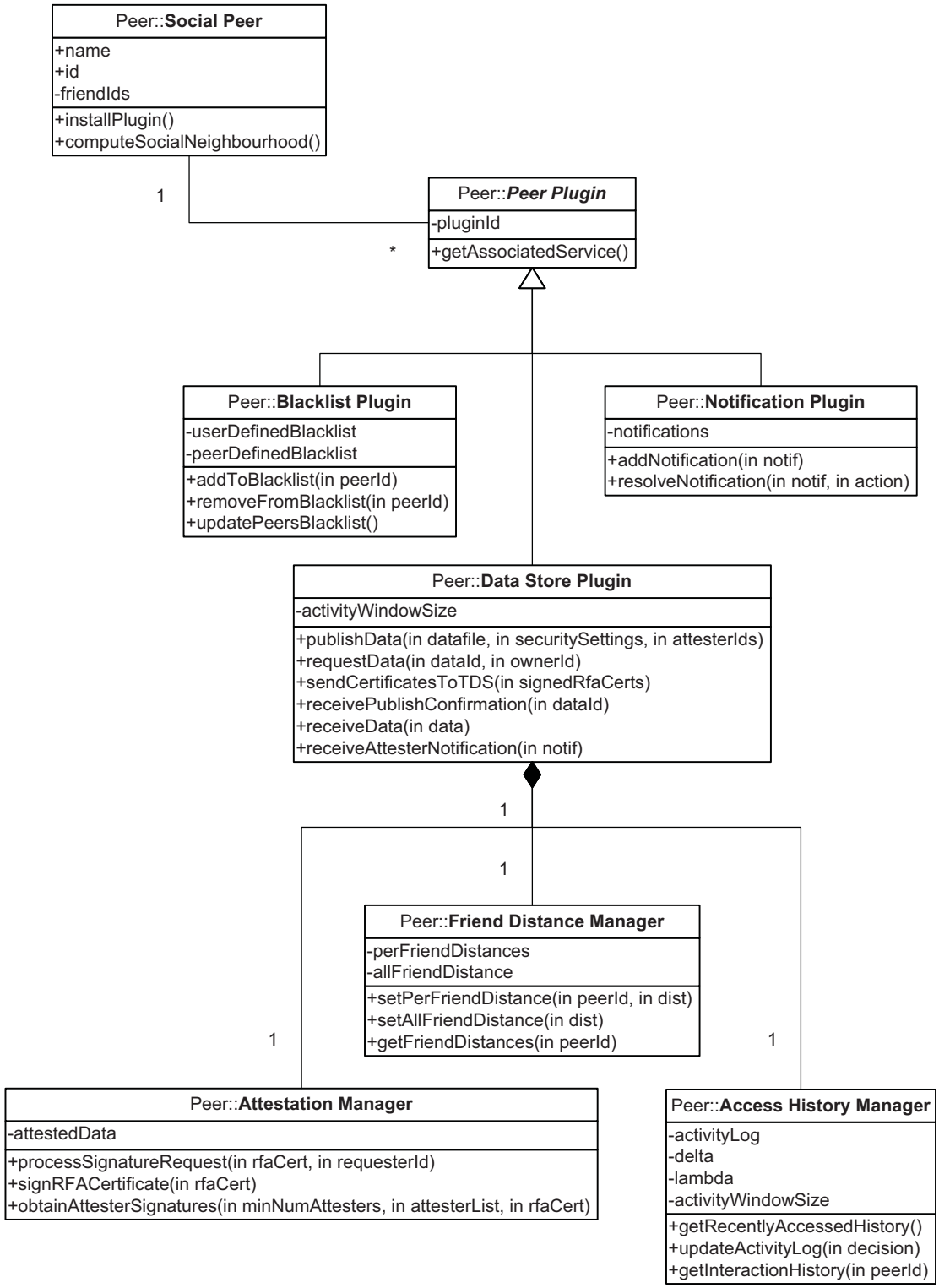

Figure A.1 Class Diagram for the Peer-side Application of the PDAC Prototype.

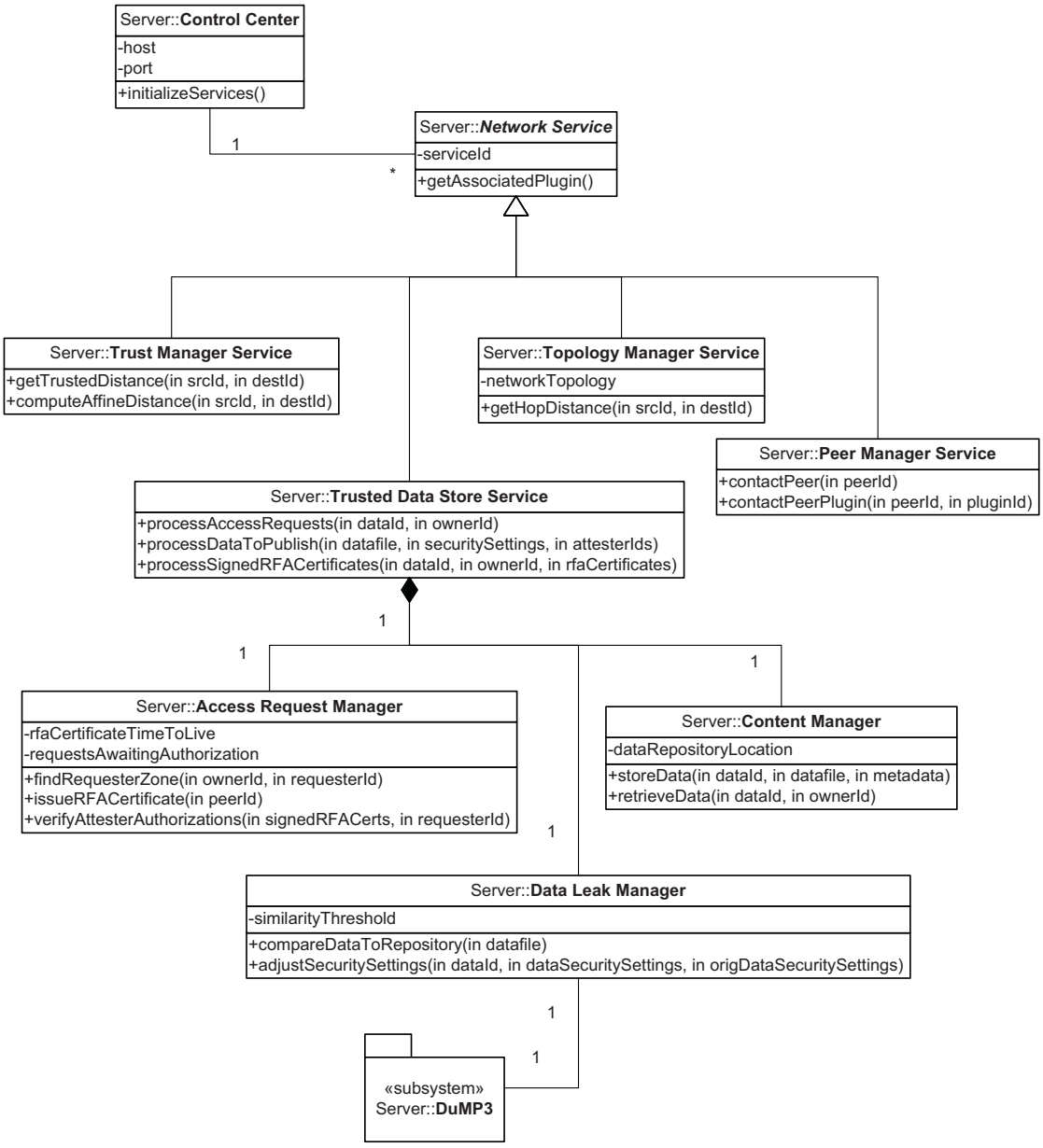

Figure A.2 Class Diagram for the Server-side Application of the PDAC Prototype.

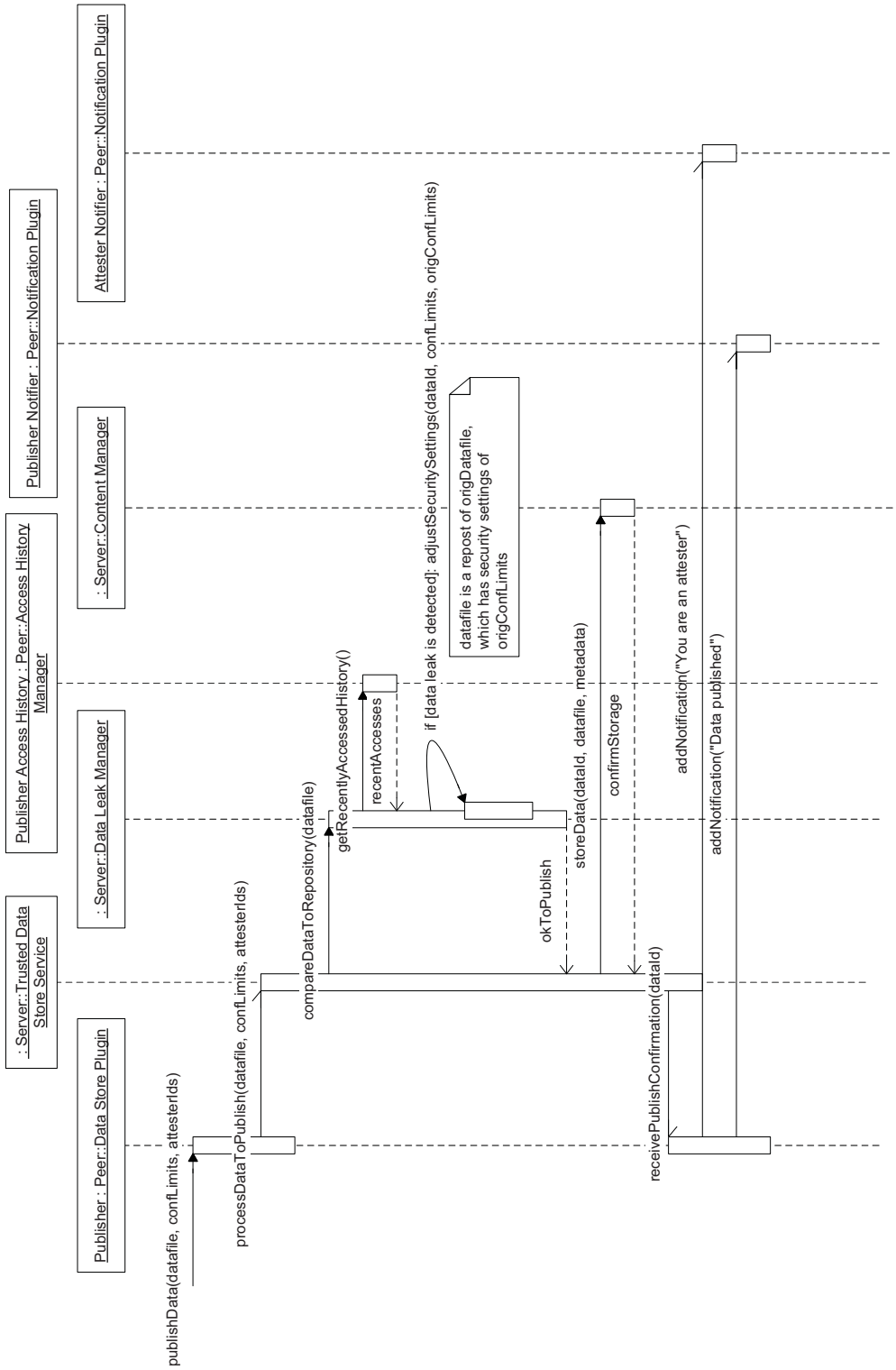

Figure A.3 Sequence diagram for publishing data on the network. In this scenario, the publisher has attempted to repost a data item that he or she has recently accessed under lower confidentiality constraints.

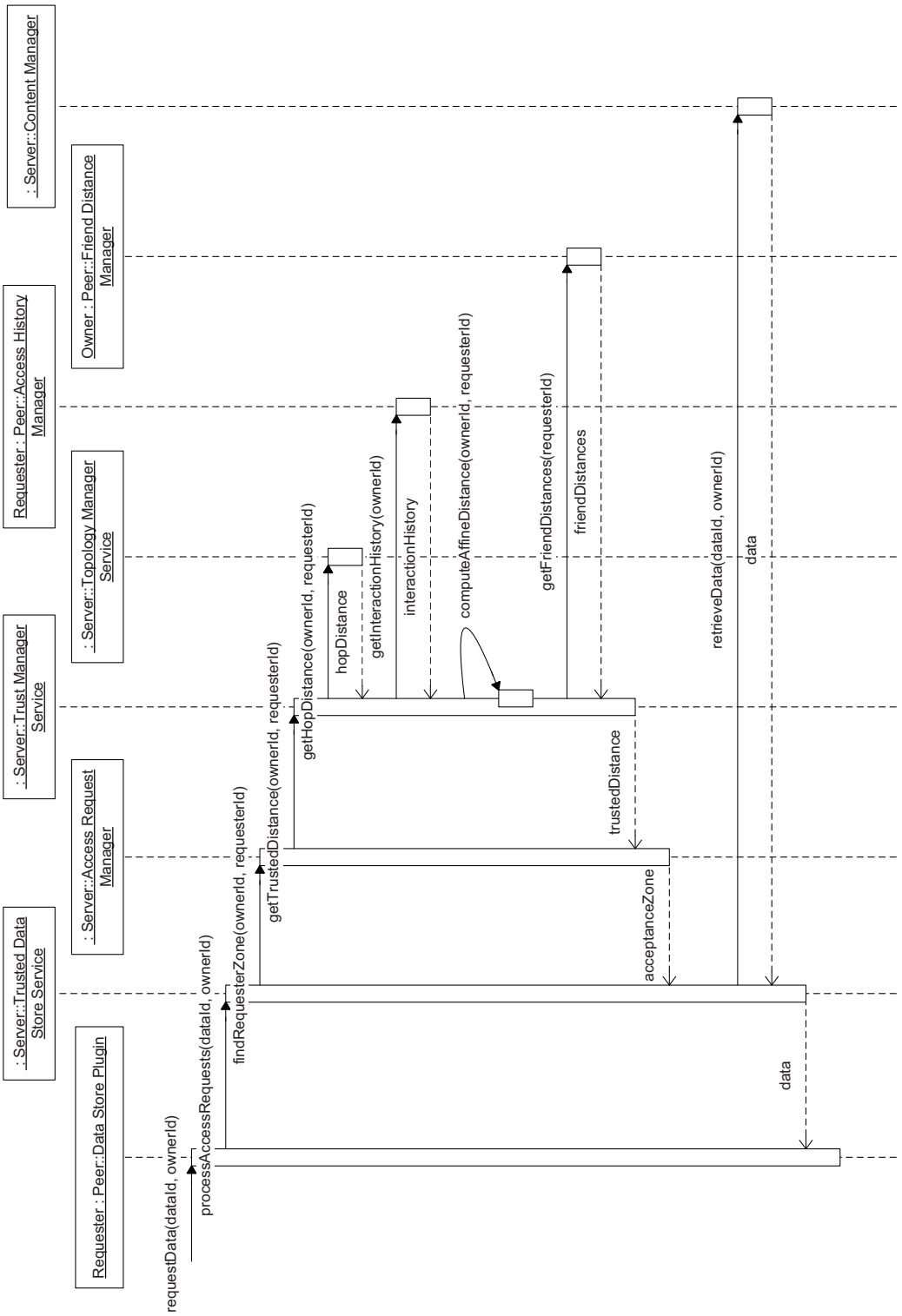

Figure A.4 Sequence diagram for requesting access to a data item on the network. This scenario has the requester falling in the Acceptance Zone.

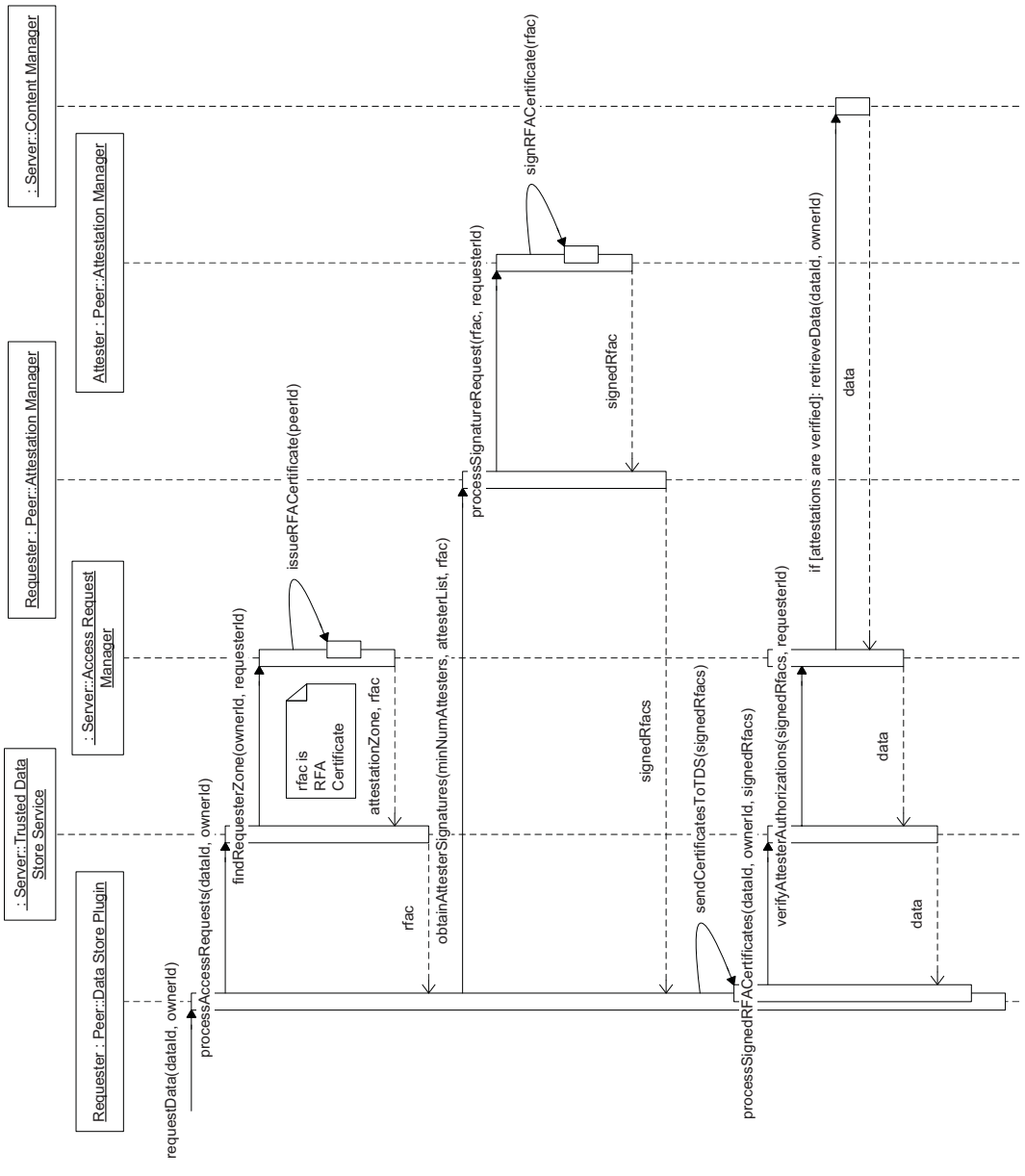

Figure A.5 Sequence diagram for requesting access to a data item on the network. This scenario has the requester falling in the Attestation Zone.

References

[1] R. Ramakrishnan and A. Tomkins, "Toward a peopleweb," *Computer*, vol. 40, no. 8, pp. 63–72, 2007.

[2] R. S. Sandhu and P. Samarati, "Access control: Principles and practice," *IEEE Communications Magazine*, vol. 32, no. 9, pp. 40–48, 1994.

[3] S. Wasserman and K. Faust, *Social Network Analysis : Methods and Applications*. Cambridge University Press, 1994.

[4] A. Mislove, M. Marcon, K. P. Gummadi, P. Druschel, and B. Bhattacharjee, "Measurement and analysis of online social networks," in *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, (New York, NY, USA), pp. 29–42, ACM, 2007.

[5] B. Carminati, E. Ferrari, and A. Perego, "Rule-based access control for social networks," in *OTM Workshops (2)*, pp. 1734–1744, 2006.

[6] B. Lampson, "Protection," in *Proceedings of the 5th Annual Princeton Conference on Information Sciences and Systems*, (Princeton University), pp. 437–443, 1971.

[7] M. Benantar, *Access Control Systems: Security, Identity Management and Trust Models*. New York, NY: Springer, 2006.

[8] D. E. Bell and L. J. LaPadula, "Secure computer systems: Mathematical foundations," Tech. Rep. MTR-2547, The MITRE Corporation, Bedford, MA, Mar. 1973.

[9] R. S. Sandhu, "Lattice-based access control models," *IEEE Computer*, vol. 26, pp. 9–19, Nov. 1993.

[10] D. Ferraiolo and R. Kuhn, "Role-based access controls," in *Proceedings of the 15th NIST-NCSC National Computer Security Conference*, pp. 554–563, Oct. 1992.

[11] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[12] R. K. Thomas and R. S. Sandhu, "Task-Based Authorization Controls (TBAC): A family of models for active and enterprise-oriented authorization management," in *Proceedings of the IFIP TC11 WG11.3 Eleventh International Conference on Database Securty XI*, (London, UK, UK), pp. 166–181, Chapman & Hall, Ltd., 1998.

[13] M. H. Kang, J. S. Park, and J. N. Froscher, "Access control mechanisms for inter-organizational workflow," in *SACMAT '01: Proceedings of the sixth ACM symposium on Access control models and technologies*, (New York, NY, USA), pp. 66–74, ACM, 2001.

[14] R. K. Thomas, "Team-Based Access Control (TMAC): A primitive for applying role-based access controls in collaborative environments," in *RBAC '97: Proceedings of the second ACM workshop on Role-based access control*, (New York, NY, USA), pp. 13–19, ACM, 1997.

[15] W. Tolone, G.-J. Ahn, T. Pai, and S.-P. Hong, "Access control in collaborative systems," *ACM Comput. Surv.*, vol. 37, no. 1, pp. 29–41, 2005.

[16] A. Bullock and S. Benford, "An access control framework for multi-user collaborative environments," in *GROUP '99: Proceedings of the international ACM SIGGROUP conference on Supporting group work*, (New York, NY, USA), pp. 140–149, ACM, 1999.

[17] M. J. Covington, W. Long, S. Srinivasan, A. K. Dev, M. Ahamad, and G. D. Abowd, "Securing context-aware applications using environment roles," in *SACMAT '01: Proceedings of the sixth ACM symposium on Access control models and technologies*, (New York, NY, USA), pp. 10–20, ACM, 2001.

[18] P. Dewan and H. Shen, "Flexible meta access-control for collaborative applications," in *CSCW '98: Proceedings of the 1998 ACM conference on Computer supported cooperative work*, (New York, NY, USA), pp. 247–256, ACM, 1998.

[19] U. Manber, "Finding similar files in a large file system," in *Proceedings of the USENIX Winter 1994 Technical Conference*, (San Fransisco, CA, USA), pp. 1–10, USENIX Association, 17–21 1994.

[20] N. Heintze, "Scalable document fingerprinting," in *1996 USENIX Workshop on Electronic Commerce*, November 1996.

[21] S. Schleimer, D. S. Wilkerson, and A. Aiken, "Winnowing: local algorithms for document fingerprinting," in *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pp. 76–85, June 2003.

[22] J. J. Hull and J. Cullen, "Document image similarity and equivalence detection," in *ICDAR '97: Proceedings of the 4th International Conference on Document Analysis and Recognition*, (Washington, DC, USA), pp. 308–313, IEEE Computer Society, 1997.

[23] S. Lin, M. T. Özsu, V. Oria, and R. T. Ng, "An extendible hash for multi-precision similarity querying of image databases," in *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, (San Francisco, CA, USA), pp. 221–230, Morgan Kaufmann Publishers Inc., 2001.

[24] G. Jomier, M. Manouvrier, V. Oria, and M. Rukoz, "Multi-level index for global and partial content-based image retrieval," in *ICDEW '05: Proceedings of the 21st International Conference on Data Engineering Workshops*, (Washington, DC, USA), p. 1176, IEEE Computer Society, 2005.

[25] A. Swaminathan, Y. Mao, and M. Wu, "Robust and secure image hashing," *Information Forensics and Security, IEEE Transactions on*, vol. 1, pp. 215–230, June 2006.

[26] J. Haitsma and T. Kalker, "A highly robust audio fingerprinting system with an efficient search strategy," *Journal of New Music Research*, vol. 32, no. 2, pp. 211–221, 2003.

[27] L. Ghouti, A. Bouridane, and M. Ibrahim, "A fingerprinting system for musical content," in *2006 IEEE International Conference on Multimedia and Expo*, pp. 1989–1992, July 2006.

[28] S.-S. Cheung and A. Zakhor, "Efficient video similarity measurement with video signature," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, pp. 59–74, Jan. 2003.

[29] A. Grässer, *DuMP3: Duplicate and similar file finder*. `http://dump3.sourceforge.net`, 2005.

[30] D. J. Weitzner, J. Hendler, T. Berners-Lee, and D. Connolly, "Creating a policy-aware web: Discretionary, rule-based access for the world wide web," in *Web and Information Security* (E. Ferrari and B. M. Thuraisingham, eds.), pp. 1–31, Idea Group Inc., 2006.

[31] T. Ryutov, L. Zhou, C. Neuman, T. Leithead, and K. E. Seamons, "Adaptive trust negotiation and access control," in *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, (New York, NY, USA), pp. 139–146, ACM, 2005.

[32] T. Ryutov and C. Neuman, "The specification and enforcement of advanced security policies," in *POLICY '02: Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02)*, (Washington, DC, USA), p. 128, IEEE Computer Society, 2002.

[33] M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu, "Negotiating trust on the web," *IEEE Internet Computing*, vol. 6, no. 6, pp. 30–37, 2002.

[34] S. Chakraborty and I. Ray, "TrustBAC: Integrating trust relationships into the RBAC model for access control in open systems," in *SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies*, (New York, NY, USA), pp. 49–58, 2006.

[35] F. Almenárez, A. Marín, C. Campo, and R. C. García, "TrustAC: Trust-based access control for pervasive devices," in *SPC 2005: Proceedings of the Second International Conference on Security in Pervasive Computing*, pp. 225–238, 2005.

[36] S. Godik and e. T. Moses, *eXtensible Access Control Markup Language (XACML)*. Organization for the Advancement of Structured Information Standards (OASIS), `http://www.oasis-open.org`, 2003.

[37] S. Proctor, *Sun's XACML implementation*. Sun Microsystems Laboratories, `http://sunxacml.sourceforge.net`, 2004.

[38] R. Sandhu and X. Zhang, "Peer-to-peer access control architecture using trusted computing technology," in *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, (New York, NY, USA), pp. 147–158, ACM, 2005.

[39] H. Tran, M. Hitchens, V. Varadharajan, and P. Watters, "A trust based access control framework for P2P file-sharing systems," in *HICSS '05: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 9*, (Washington, DC, USA), p. 302.3, IEEE Computer Society, 2005.

[40] W. Adams and I. Davis, N.J., "Toward a decentralized trust-based access control system for dynamic collaboration," in *Proceedings of the 2005 IEEE Workshop on Information Assurance and Security*, pp. 317–324, 2005.

[41] N. Dimmock, A. Belokosztolszki, D. Eyers, J. Bacon, and K. Moody, "Using trust and risk in role-based access control policies," in *SACMAT '04: Proceedings of the ninth ACM symposium on Access control models and technologies*, (New York, NY, USA), pp. 156–162, ACM, 2004.

[42] R. Bhatti, E. Bertino, and A. Ghafoor, "A trust-based context-aware access control model for web-services," in *ICWS '04: Proceedings of the IEEE International Conference on Web Services*, pp. 184–191, June 2004.

[43] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor, "A generalized temporal role-based access control model," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 4–23, Jan. 2005.

[44] P. Bonatti, C. Duma, D. Olmedilla, and N. Shahmehri, "An integration of reputation-based and policy-based trust management," in *Semantic Web Policy Workshop in conjunction with 4th International Semantic Web Conference*, Nov. 2005.