

An Access Control Scheme for Protecting Personal Data

Wilfred Villegas, Bader Ali, and Muthucumaru Maheswaran
School of Computer Science
McGill University
Montreal, QC H3A 2A7, Canada
{wville1, bali2, maheswar}@cs.mcgill.ca

ABSTRACT

This paper presents a new *personal data access control* (PDAC) scheme that is inspired by the protection schemes practiced in communities for sharing valuable commodities. The PDAC users are assumed to be members of an online social network such as `facebook.com`. The PDAC computes a “trusted distance” measure between users that is partly based on hop distance on the social network and an affine distance derived from experiential data. Based on the trusted distance, the social network is divided into three zones: accept, attest, and deny. Requests from users in the accept zone (closest to the data origin) are accepted unconditionally while the requests from the deny zone (furthest from the data origin) are rejected outright. Requests from the attest zone need additional authorization to get access. In addition to protecting first accesses, PDAC tracks reposts to minimize data leaks (i.e., spread of data beyond the limits set by the data originator). The details of the PDAC scheme are presented in this paper along with an analysis of its properties. We implemented PDAC on a social network emulator to demonstrate its viability. The performance of certain PDAC functions were examined using simulations driven by portions of social graphs obtained from `myspace.com`.

1. INTRODUCTION

People are creating digital content such as home movies, photo albums, social network profiles, blogs, and wiki pages at a rapidly increasing rate. Such user-generated content is creating many new challenges in archiving, indexing, searching, browsing, and sharing [19]. This paper is concerned with one of the key problems – sharing.

Consider an example where Alice wants to share her growing collection of digital content with her friends. Alice wants to limit the access to sensitive data to her trusted friends while permitting open access to the rest of the data in the collection. To differentiate among the users seeking access, Alice should be able to identify the users and compute the trust she has with them. Another important concern in data

sharing is privacy and a lot of it stems from the extent of control Alice has over the sharing process [2].

The simplest way for Alice to control the access to her digital library is to mark unrestricted data objects as *public* and restricted data objects as *private*. For private data objects, Alice could create a list of users and the types of access they have for the objects. While this approach provides Alice with a high degree of control on how her data is shared, the time Alice spends managing access can scale up as a product of the number of data objects, number of different contexts, and number of users. For Alice to continue producing data and share it with others, we need to develop protection techniques that incur low management overheads.

Online social networks such as `facebook.com` and `myspace.com` are popular approaches for representing friendships. Assume that Alice and her friends use `facebook.com` to connect to one another. Clearly, information about Alice’s friendship circle from `facebook.com` could form the basis for a protection scheme for her personal data. For instance, we could use the hop distance to categorize people connected to Alice as *direct friend* (1 hop away from Alice) and *friend-of-a-friend* (2 hops away from Alice). Once such a categorization of people is made, we could institute sharing policies for different data objects in terms of hop distances from Alice. For example, Alice’s family photo album might be shared only up to a 1 hop distance and a photo collection from an office retreat might be shared up to a 2 hop distance. While the above strategy to control sharing is straightforward to maintain, it assumes *all* friendships are *equal*.

Many practical sharing situations require mechanisms that provide fine-grained protection. For instance, Alice might have office and family friends at hop 1 on the social network. However, depending on the object being shared, she might want to limit access to only one of the groups. With fine-grained protection, the complexity of access management can rise rapidly. Restricting access to data in social networks is very similar to restricting traffic on the Internet by implementing access policies using firewalls. Previous studies [3, 25] on corporate firewalls have shown that complexity is one important cause of errors that have compromised the defensive capabilities of the firewalls. If corporate firewalls that are configured and managed by professionals are susceptible for complexities, we can expect complexity to have a much pronounced effect on data-sharing in online systems (e.g., social networks and online storage systems), where policies are implemented by ordinary users.

In this paper, we propose *personal data access control* (PDAC), a novel, cooperative strategy for protecting personal data that is inspired by the social network centered approach that people have used for exchanging valuable commodities [15, 17]. One of the key underlying objectives of PDAC is to develop a flexible access control scheme for online data repositories that require minimal user intervention. The PDAC introduces three protection zones: accept zone, attestation zone, and deny zone. To use PDAC, Alice defines the protection zones for her data collection and maps all users from her social neighborhood into one of the zones. The mapping process is not exclusively done by Alice because this would require Alice to be all knowing. Instead, Alice sets the baseline sharing parameters and expect the community (i.e., the social network) to provide significant help in mapping the users to the different zone. The users falling into the accept zone get unconditional access to Alice’s data objects. Conversely, requests from users falling into the deny zone are rejected outright. Requests from users falling in the attestation zone are validated on a per-request basis. This paper describes the complete PDAC scheme and demonstrates that PDAC provides a flexible protection mechanism that is user friendly and requires low management effort.

We evaluate PDAC in two ways: (i) analyze the user effort complexity to show PDAC can work on very large scale networks and (ii) use simulations to compare PDAC with a hop-based scheme and measure the errors caused by malicious users. Further, we implemented the PDAC on top of a social network emulator that faithfully represents existing social networks. Our evaluations demonstrate that PDAC is applicable in the context of existing social networks.

Section 2 discusses the requirements that should be met by any protection scheme for personal data. Section 3 describes the models we assume for trust management within social networks. The personal data access control strategy is presented in Section 4. The scalability and other performance parameters of PDAC are examined in Section 5. We briefly discuss the implementation of PDAC on a social network emulator in Section 6. Related work is discussed in Section 7.

2. PERSONAL DATA PROTECTION

2.1 Issues with a Naive Protection Scheme

To illustrate the requirements of personal data protection, we start off with a simple access control scheme, where Alice sets access limits based on connection distances on a social network. For example, for a personal photo album that Alice posts on an online storage site, she would specify that it should be accessible only to her direct friends (i.e., people one hop away from Alice on the social network). For another data set that contains less sensitive data and that requires wider distribution (e.g., a business plan), Alice can include friend-of-friends (i.e., people within two hops from Alice on the social network). While this scheme is very simple and is already implemented by several existing social networks (for example, both `facebook.com` and `friendster.com` support up to 2 hops), it has several drawbacks. One in particular is the inability to easily exclude *some* friends and include *some* friend-of-friends in the sharing process. This drawback can be addressed by including white or black listing capabilities, where white lists define a set of users who should be provided

access, and black lists define a set of users who should be denied access (for example, `facebook.com` supports both).

Although the simple access control scheme sketched above or variations of it has been implemented in actual social networks, it has many drawbacks. Many of these drawbacks stem from the following assumptions made by the simple schemes that are not always applicable.

1. **All friends are equal.** Access control schemes that use the hop distance to categorize friends assume that all friends at a particular hop distance are equal. While this makes access control simple, we often need the flexibility of differentiating among the friends so that for certain data objects Alice could be selective among her friends.
2. **Omniscient users.** Access control schemes that expect users to define black and white lists to explicitly deny and allow accesses to data objects assume that users are all knowing about their social neighborhood and able to make the appropriate protection decisions. With large and dynamic social neighborhoods (friends and friend-of-friends) such an assumption is not practical.
3. **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 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.

2.2 Requirements of Personal Data Protection

A scheme meant for personal data protection should provide the user *full control* over how her data is shared. A personal data collection can have variety of different data and the sharing requirements can be diverse. Therefore, the protection scheme should be *flexible* and capable of operating at different data granularities. This flexibility requirement has various aspects including: applying different sharing criteria for different friends, changing sharing conditions on a data object basis (e.g., increasing the confidentiality level of an already published data object), and changing sharing conditions on a friend basis (e.g., decreasing the trustworthiness of a friend and consequently limiting her access to user’s data). While flexibility is essential in protection schemes, it 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 in its eventual acceptance in the user community. One way of retaining the flexibility and *reducing* the *user effort* required by the scheme is to enable *collaborative* decision making. The collaborative decision making for access control for Alice can take different forms: learning from Alice’s past access control actions, learning from past actions of the community of users within a social neighborhood of Alice, learning from the past actions of like-minded set of users within Alice’s social neighborhood, or a combination of the above.

Another important requirement of personal data protection is *accessibility* to data. For instance, Alice would like to know the friends or friend-of-friends who can access the new photo album she is posting with a particular confidentiality setting. Such prediction of accessibility can be used to in-

teractively shape the confidentiality settings for important personal data collections.

2.3 Motivation for a Social Networking Approach

Social networking and analysis of such networks has been studied in sociology, economics, and political science for several decades [24]. One thrust of work from economics that has significantly inspired our work is microfinance [11, 17, 15] where social connections within communities are leveraged to minimize the risk in lending. It has been observed (particularly in developing countries) that the lender (e.g., a bank) does not have the resources to directly monitor borrower behavior. As a low cost alternative, the microfinance model enlists the help of peers from the same social network as the borrower to monitor and persuade the borrower to abide by the terms of lending. A field study by Dean Karlan [15] indicates that the structure of the social graph is an important factor in influencing the repayment rates. In many ways, the personal data protection problem is similar to the microfinance problem. Like a lender in microfinance wanting to protect her capital, a data object owner wants to protect her data by enlisting the help the social network peers.

While the online social networks have created their own share of privacy and security issues, they provide ideal frameworks to capture the social structure in a large community. The main thrust of this paper is to exploit the social structure within these networks to create a new data protection scheme.

3. ASSUMPTIONS AND SYSTEM MODEL

This paper makes the following assumptions regarding the system setup. All users belong to a centrally maintained social network such as `facebook.com` or `myspace.com`. The social network is connected and structured as a single giant component. 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 users from getting cheated by fake friends and theft of user credentials.

The access control scheme such as the one presented here 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.

3.1 Modeling 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. We propose to quantify the trust between two users using a *trusted distance* measure. The trusted distance measure is similar to the hop (link) distance between users on current online social networks. For instance, direct friends (1 hop friends) are considered more trusted than friend-of-friends (2 hop friends).

The trusted distance measure is a generalization of the hop distance. The trusted distance is a real value, unlike hop count which is an integer value. Also, trusted distance can include hop distance along with other factors relevant to inter-personal trust.

3.2 Confidentiality Policies

One of the major objectives of personal data protection is enforcing confidentiality constraints. The protection mechanisms should allow the users to specify confidentiality policies that are tailored for the different data objects. We need to deal with two types of concerns with regard to confidentiality constraints: initial access to data objects and reposting of data objects.

3.3 Data Storage

We assume the availability of a centralized *trusted data store* (TDS) that can be used by all users to store and retrieve data objects (for a distributed design of the TDS, please see [23]). The TDS is composed of four separate components: the *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 4.5. 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 (i.e., whether access requests were accepted or rejected). This access request log is used by the Trust Manager in the trust computation process as described in the next section. In a future work, we will discuss how this scheme can be extended to use multiple data stores to provide fault tolerance to the system.

4. PERSONAL DATA ACCESS CONTROL

Personal Data Access Control (PDAC) is designed to protect personal data that users post on social networks and other online storage systems. When Alice posts a data object, she specifies the confidentiality constraints using a trusted distance measure. As shown in Figure 1, the confidentiality specifications divides the people seeking access into three zones with respect to Alice: highly trusted friends, marginally trusted friends, and untrusted people. Requests from highly trusted friends are automatically honored while the requests from marginally trusted friends are honored only if the attestors designated by Alice are willing to undersign the trustworthiness of the requester. Requests from untrusted people are automatically rejected.

4.1 Confidentiality Limits and Specification

The PDAC protocol implements the confidentiality requirements in two stages. The first stage uses two thresholds on the trusted distance from Alice to define the three zones. For a data object d_k owned by Alice, the *accept limit* ($c_a(\text{Alice}, k)$) is the largest trusted distance for which an access request is honored unconditionally. Similarly, the *deny limit* ($c_d(\text{Alice}, k)$) is the smallest trusted distance for which an access request is rejected. If the trusted distance to a requester for data object d_k is in $(c_a(\text{Alice}, k), c_d(\text{Alice}, k))$,

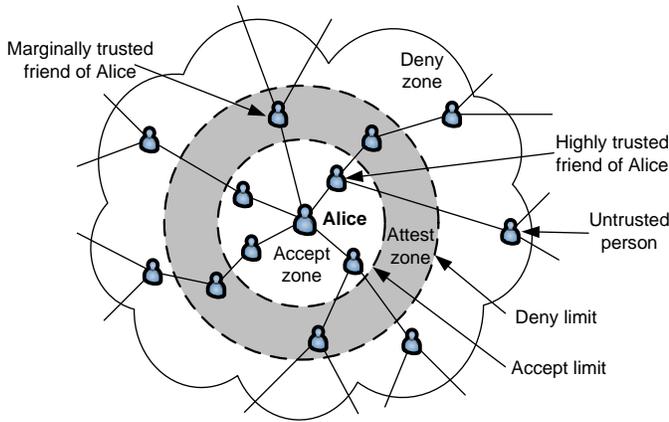


Figure 1: Illustration of data protection zones for one of Alice's data collections with user mappings.

the request should be authorized by the attesters designated by Alice. The second stage computes the trusted distance for a user from Alice and places the user in the appropriate zone.

For the purposes of this paper, we assume 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), *very good friend* by a number less than that used for a good friend (for example 0.8), *good friend-of-a-friend* by a number less than 2.0 (for example 1.8), and so on. Ideally, to keep the system user-friendly we need mechanisms that will accept confidentiality specifications in fuzzy mnemonic labels such as **very good friend** or **good friend**. With a mnemonic labeling scheme, Alice can include a specification such as [**good friend**, **very good friend-of-a-friend**] with a data object she posts online. This specification means that Alice wants to unconditionally provide access to someone who is at least a good friend and reject anyone who is at least not a very good friend-of-a-friend. The proof-of-concept prototype we discuss in this paper does not handle mnemonic labels.

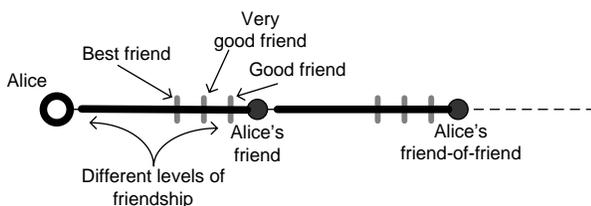


Figure 2: Illustration of trust distance labels.

To map the users into the different zones defined by Alice, the PDAC protocol computes the trusted distance to each user from Alice. The PDAC protocol employs a cooperative strategy for computing the trusted distances. As shown below, we deduce trust measures from prior user activities as registered in the activity log. The entries in the activity log can be tagged to indicate the context under which they took place. Using the tagging information, we can filter

past activities that are not relevant to the current context. Consider an example where Bob is seeking access to data objects posted by Alice that are categorized as *work*. If Bob is a family friend of Alice and all prior activities are logged with the tag *family*, the contribution to the trusted distance between Alice and Bob from prior activities in the context of *work* should be zero. Although the scheme presented in this paper can be easily extended to handle multiple contexts, for the rest of the paper we assume a single context.

4.2 Trusted Distance Computation and Use

We assume all users are connected by a social graph. Let x and y denote two users on the social graph. The *hop distance* $-d_{hop}(x, y)$ is defined as the smallest number of hops that separate x and y on the social graph. The hop distance is a fairly static parameter that can change only 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 context dependence and other variabilities introduced by interactions, we introduce a measure called *affine distance* $-d_{aff}(x, y)$. For the purposes of this paper, we say $d_{aff}(x, y)$ is solely dependent on how x or other users on x 's social neighborhood have honored y 's access requests in the past.

While the hop and affine distances defined above indicate trust between users, they are not suitable for indicating sudden changes in protection requirements. To make rapid changes in protection requirements, we introduce a *friend distance* parameter. The friend distance parameter can have two components: an *all friend distance* and *per-friend distance*. The all friend distance $-d_{afdst}(x) \geq 0$ is applied between user x and all other users. Normally, this parameter is set to 0. A positive value indicates that user x wants severely restricted sharing for all her data. Similarly, the per-friend distance $-d_{pfdst}(x, y) \geq 0$ is applied between user x and user y and this is set to 0 by default. A positive value here indicates that user x wants to severely restrict data sharing with user y .

Unlike the hop and friend distances, the affine distance is not directly set by the user. It is computed by mining past user activities. We present two approaches for computing $d_{aff}(x, y)$ between x and y . We assume that the Access Control Manager of the TDS maintains activity logs for users' activities for a pre-defined time window (e.g., one week or one month). For instance, the activity log for user y maintains information on all the requests y made and the outcomes of those requests. An entry for a request would indicate the data object requested by y , identity of the original poster of the data object, time of request, tags describing context, and outcome of the request. Suppose y 's activity log has q_y total number of requests over the current activity window. Of the q_y requests, suppose a_y were accepted and r_y were rejected. The history information is compressed into a single value s_y , the long term success rate, by computing an aggregate success rate. The success rate is scaled by a factor which depends on the unique number of data posters k that have accepted requests made by y . The value of s_y is in $(-1.0, 1.0)$.

$$s_y = \left(\frac{r_y - a_y}{q_y}\right) \left(\frac{1}{1 + e^{-k/\alpha + \beta}}\right) \quad (1)$$

In the equation above α and β are system parameters used

to control the scaling factor.

The simplest way for x to compute $d_{aff}(x, y)$ is to use 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 and a_{yx} be the number of accepts and r_{yx} be the number of rejects. Let λ be a tuning parameter in $[0, 1]$ and Δ be an arbitrary small number. To obtain the $d_{aff}(x, y)$, the long term success rate of y given by s_y is combined with the success rate of y for x 's data objects as shown in the following equation.

$$d_{aff}(x, y) = \lambda s_y + (1 - \lambda) \left(\frac{r_{yx} - a_{yx}}{q_{yx} + \Delta} \right) \quad (2)$$

In our current scheme for finding $d_{aff}(x, y)$ we only include y 's interactions with users from x 's social neighborhood when calculating the long term success rate s_y . This method is motivated by the assumption that polling x 's social neighborhood provides additional information on y that can be used to derive a good estimate for $d_{aff}(x, y)$. This method can be further improved by considering a like-minded (user making similar access control decisions) subset of x 's social neighborhood.

Once the affine distance $d_{aff}(x, y)$ between x and y is computed, we add it to the hop distance $d_{hop}(x, y)$ and friend distances $d_{afdst}(x)$ and $d_{pfdst}(x, y)$ to obtain the trusted distance $d_{trust}(x, y)$.

$$d_{trust}(x, y) = d_{hop}(x, y) + d_{aff}(x, y) + d_{afdst}(x) + d_{pfdst}(x, y) \quad (3)$$

In the above equation, it is important to make a few observations. The $d_{hop}(x, y)$ assumes discrete values such as 1 and 2 representing a friend and a friend-of-a-friend, respectively. The $d_{aff}(x, y)$ values are in the range $(-1.0, 1.0)$. The friend distances $d_{afdst}(x)$ and $d_{pfdst}(x, y)$ are greater than or equal to 0. This means the trusted distance of a friend-of-a-friend cannot become less than or equal to that of a friend in the above scheme.

The value of $d_{trust}(Alice, y)$ determines which zone user y belongs to with respect to Alice. Taken together with the confidentiality limits set by Alice on her data objects, the trusted distance determines whether data object d_k belonging to Alice is accessible to user y as summarized by the following rules:

1. $d_{trust}(Alice, y) \leq c_a(Alice, k)$: User y is considered highly trusted by Alice and is automatically given access to the data object. This is called the *Acceptance zone*.
2. $c_a(Alice, k) < d_{trust}(Alice, y) < c_d(Alice, k)$: User y must be *attested* to access the data object. This is called the *Attestation zone*.
3. $c_d(Alice, k) \leq d_{trust}(Alice, y)$: User y 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 4.3. Note that because a user's activity log maintained at the Access Control 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.

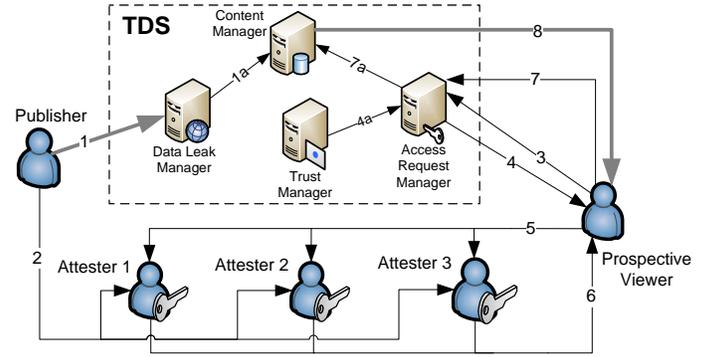


Figure 3: The steps involved in publishing and requesting data. To publish a data item, the publisher sends it to the TDS (1) and notifies the attesters (2). Before accepting the data item for publication, the TDS performs a data leak analysis and adjusts the confidentiality limits if needed (1a). A 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).

4.3 Attestation Process

In addition to the confidentiality limits, Alice must specify a list of n attesters when she publishes a data object. The attesters are members of the social network whom Alice considers trustworthy. When a request falls into the attestation zone – meaning the requester does not have the complete trust of Alice – the attesters are asked to undersign the request. If k out of n attesters undersign the request, the Access Request Manager will grant the requester's access. Optionally, Alice could indicate to the attesters the conditions that should be fulfilled by the requester before the attesters authorize the request. In this paper, we specify only one type of condition – hop distance to the requester from the attester. For example, Alice can specify that the attesters should limit authorizations to requesters that are within 2 hops of them.

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 family photo album. Similarly, for a business document authored by Alice, she could enlist trustworthy co-workers as attesters.

When a client requests a protected data object from the TDS, the Access Request Manager assigns the client a certified *request of attestation* (RFA) document. The RFA lists the n attesters and the minimum number of attestations required for the Access Request Manager to approve the request. Once the client is successful in obtaining at least k attestations, it can submit them to the Access Request Manager. The attestations should include a digest of the RFA to prevent replay of old attestations. The RFA can include an expiry time to force the client to complete the attestation process within a limited time frame.

4.4 Changing Access Limits

Because the primary goal of PDAC is personal data protection, it is important to provide the user the flexibility of changing the access control parameters even after posting the data objects. Below we discuss the changes that are supported by PDAC.

1. **Restrict access for a single data object:** In this case, Alice wants to restrict access for a particular data object she has previously posted on the TDS. With the original publication, Alice specified two thresholds that correspond to accept and deny limits. These values can be changed at the Content Manager to reflect the new access control requirements.
2. **Restrict access for all data objects:** In this case, Alice wants to restrict access to all data objects she has published on the TDS. This can be easily accomplished by increasing the all friends distance t_{afdst} .
3. **Restrict access for a particular user:** In this case, Alice wants to prevent a given user (e.g., Carol from accessing some of her data objects. Such a restriction can be setup by increasing the per friend distance t_{pfdst} that corresponds to Carol.
4. **Restrict access via attestation for data objects:** The attesters examine requests from requesters that have trusted distance falling between the two limits. When Alice publishes a data object, she associates an attestation condition with it. The conditions are stored with the data object at the Content Manager in the TDS and are noted down in the RFA issued by the Access Request Manager. To change the attestation condition, Alice should retrieve the records associated with the data object in the Content Manager and change the attestation conditions. For example, Alice could specify that attesters should verify that the requester is within 1 hop instead of the default 2 hops.

4.5 Data Reposting and Information Leakage

The PDAC protocol components 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. Suppose Alice posts a personal photo album on the TDS with friend-of-a-friend deny limit. A friend of Alice, Carol, 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 dictate the confidentiality constraints that should be used by Carol. If Carol reposts the same album with a friend-of-a-friend deny limit, users who do not meet the original trusted distance requirement can access Alice’s photos. This clearly violates the original confidentiality requirements.

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 that can be attached with a data object. For instance, when Carol submits a data object to TDS for posting, the Data Leak Manager performs a similarity analysis of that object against the data objects to which Carol

recently gained access. If the data object presented by Carol is sufficiently different from those data objects that she recently accessed (i.e., data object level similarity measure is less than a pre-defined threshold), the data object is accepted for posting. Otherwise, the confidentiality limits are modified as follows.

For each data object d_i posted on the Content Manager, the Data Leak Manager extracts feature vectors using document similarity measurement techniques. Because document similarity measurement is beyond the scope of our main research thrust we use existing techniques developed for plagiarism detection on student courseworks [20], virus fingerprinting [18], and copyright violations [21]. Using the n features the Data Leak Manager extracts for each data object, it computes a feature match score $0 \leq m_{ij} \leq 1$ between two objects d_i and d_j as the fraction of matching features.

Consider the case where Carol is submitting data object d_i to TDS. A similarity analysis by the Data Leak Manager for data object d_i reveals a set of data objects \mathcal{D}_m such that for each $d_k \in \mathcal{D}_m$ we have a feature match score m_{ik} that is above the similarity threshold τ used by the Data Leak Manager. Further, using access logs from the Access Request Manager, the Data Leak Manager determines a subset of data objects in \mathcal{D}_m that has been accessed by Carol. Let this subset be \mathcal{D}_{ma} . We can conjecture that the similarity (indicated by a high feature match score) between the object posted by Carol and the objects in \mathcal{D}_{ma} is because Carol has reused those objects wholly or partly in her submission.

Suppose u_l is the owner of a matching data object $d_k \in \mathcal{D}_{ma}$ and has specified $c_a(l, k)$ and $c_d(l, k)$ as the accept and deny limits, respectively for the object. Let the trusted distance from u_l to Carol be given by $d_{trust}(l, Carol)$ and hop distance along the social graph be given by $d_{hop}(l, Carol)$. The PDAC allows the original poster of a data object (i.e., u_l in this case) to choose between a *relaxed* and *strict* privacy setting. With relaxed privacy, the accept limit Carol can specify should be less than or equal to $c_a(l, k) - d_{trust}(l, Carol)$ and the deny limit should be less than or equal to $c_d(l, k) - d_{trust}(l, Carol)$. Similarly, with the strict privacy setting, the accept limit Carol can specify should be less than or equal to $c_a(l, k) - d_{hop}(l, Carol)$ and the deny limit should be less than or equal to $c_d(l, k) - d_{hop}(l, Carol)$. In Section 5, we derive certain properties of the strict and relaxed privacy settings.

4.6 Example Scenario

Consider the following scenario. We assume $\lambda = 0.4$ and $\Delta = 0.001$. Alice wishes to share an album of personal photographs p 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 $c_a(Alice, p) = 0.5$ and a deny limit of $c_d(Alice, p) = 2.5$. She also assigns her friends Ivan, Trent, Pat, and Vanna 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. Bob, a mutual friend of Pat and Vanna, sends a request to the Access Request Manager of the TDS to view the pictures. Since Bob has had no previous interactions with Alice, $d_{trust}(Alice, Bob) = d_{hop}(Alice, Bob) = 2$, and the Access Request Manager issues Bob an RFA document. Bob obtains authorizations

from both Pat and Vanna since they both know him. Bob then submits the authorizations to the Access Request Manager which verifies that Bob has satisfied the attestation requirements. Bob has now gained access to Alice’s photo album. At this point, $d_{aff}(Alice, Bob) = -0.599$, which gives $d_{trust}(Alice, Bob) = 1.401$. This value captures the trust gained by Bob through his attestations. Oscar, a friend of Bob, also wishes to view the photo album. Although Bob added Oscar as a friend on the social network, he has denied Oscar access to a few of his data items, so $d_{aff}(Bob, Oscar) = 0.6$, resulting in $d_{trust}(Bob, Oscar) = 1.6$. Oscar’s access request is denied because he has no previous interactions with Alice and $d_{trust}(Alice, Oscar) = 3$.

Suppose now that Bob chooses to republish the Alice’s photographs, to which he just gained access. Bob attempts to publish the same photographs using an accept limit of $c_a(Bob, p_{copy}) = 1$ and a deny limit of $c_d(Bob, p_{copy}) = 3$. The Data Leak Manager of the TDS detects that Bob is attempting to republish Alice’s photo album (which Bob was just granted access to) under weaker confidentiality limits. To preserve the original limits set by Alice, the Data Leak Manager modifies the confidentiality limits submitted by Bob. Since the original data had a privacy setting of *strict*, the modified values are $c_a(Bob, p_{copy}) = 0$ for the accept limit and $c_d(Bob, p_{copy}) = 0.5$ for the deny limit. This ensures that Oscar will still be unable to view the photographs, thus preserving Alice’s original confidentiality limits. Note also that because Bob was originally attested to view the photographs, anybody wishing to view Bob’s copy of the photographs must also be attested.

5. ANALYSIS OF PDAC

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: dual classed, hop based, and list based. In dual classed, a data object is tagged as *private* or *public*. The private data objects are restricted to the owner or a small set of users selected a priori by the owner. The hop-based scheme discussed in Section 2.1 provides finer control over the access control process than the dual-class scheme. The hop-based scheme exploits the structure in the social graph that interconnects the users. In the list-based scheme, a black list is associated with each object that enumerates the users who should be denied access to it.

Let m be the total number of data objects and n be the total number of users in the system. In the dual-class scheme, the user should at least tag each object as public or private. Therefore, a dual-class 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 requires at least m user actions (same as dual-class). In the list-based scheme, the owner of a data object enumerates the users that should have access to the object. The lists can be either white or black. With white or black listing, default rules can be used to reduce the number of user specifications. For instance, a white listing scheme would imply “all other users” not enumerated

in the list are denied access. Therefore, user i requires s_i actions to set the parameters of the white or black list, where s_i is number of users from the global population that user i cares to enumerate in her list (e.g., the size of the social neighborhood). Suppose s_n is the average size of the social neighborhood over the full system, then the total number of user actions required by a list-based scheme is ms_n .

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 allows users to black list other users in the network by setting the *per friend distance* to non-zero values. Black listing in PDAC is user-specific, unlike in the list-based scheme where black listing is specific to each data object. Suppose user i marks b_i other users as “bad” (i.e., wants to avoid sharing her data). 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 $2m + nb_n$. We assume the number of malicious users in a social network to be low, so the $2m$ term dominates.

5.2 Simulation Results

In this section, we show the results from the simulations we performed to evaluate PDAC and compare it with a hop-based access control scheme. In hop-based access control the access decision is based exclusively on the hop distance, so a requester is given access if his distance from the owner is within the hop limit set by the owner. To setup the simulation, we used portions of a social graph obtained from myspace.com. We then used a request generator (an oracle) to generate a set of requests along with the expected access outcomes. The set of requests generated by the oracle was then 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. We used three metrics 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 is defined as the fraction of requests that were given access by each scheme but were denied access by the oracle, and false negatives are defined as the fraction of requests that were denied access by each scheme but given access by the oracle. We also introduced malicious users into the network to evaluate the effectiveness of PDAC in detecting and denying access to these 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 (random) or hop-based discrete. When the uniform (or random) distribution is used for request generation, an access request can originate from any hop distance with equal probability. When a hop-based discrete distribution is used for request generation, the probability of an access request emerging within a particular hop-count is given by the distribution. For example, the hop-based discrete distribution (0.5,0.35,0.08,0.04,0.02,0.01) specifies the probability values for requests to originate from hops (1,2,3,4,5,6). That is, the probability of request originating at hop 1 or less is 0.5, hop 2 or less is 0.85, hop 3 or less is 0.93 and so on. For the hop-based discrete distribution we define three distinct probability density functions. The discrete steep pdf is de-

		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	0.751	0.117	0.133	0.008	0.697	0.241	0.062	0.615
discrete shallow	discrete steep	0.681	0.150	0.169	0.013	0.669	0.270	0.062	0.539
discrete shallow	discrete shallow	0.677	0.190	0.133	0.031	0.529	0.463	0.008	0.914
random	discrete steep	0.928	0.028	0.044	0.011	0.950	0.003	0.047	0.005
random	discrete shallow	0.894	0.025	0.081	0.011	0.863	0.093	0.044	0.144

Table 1: Comparison of simulation results between PDAC and hop-based scheme.

defined as (0.5,0.35,0.08,0.04,0.02,0.01) for hops (1,2,3,4,5,6) respectively. Similarly, discrete shallow pdf is defined as (0.3,0.3,0.3,0.05,0.03,0.02) and discrete shallower pdf is defined as (0.25,0.25,0.25,0.15,0.07,0.03). To generate a request a source user is first selected at random. The requester is then chosen according to either the random, discrete shallow, or discrete shallower distribution. Similarly the outcome of each request is determined based on either the discrete steep or discrete shallow distribution.

For the purposes of the simulations, the parameters of both PDAC and hop-based were fixed for each outcome distribution. 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.

Table 1 shows the results for different runs while having 10% malicious users placed randomly in the network. In almost all of the simulations, the PDAC had a higher success rate with lower false positives and higher false negatives. This shows that PDAC reduces the incidence of unauthorized data accesses by at least half in most cases. The higher incidence of false negatives in PDAC means that it is more conservative when granting access, indicating a more protective mechanism. For the (random, discrete steep) case the hop-based method has better values for almost all categories. One of the reasons the hop-based scheme performs well for the (random, discrete steep) combination 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 neighborhood with the random request generation. Given the discrete steep outcome distribution, the oracle will reject requesters that are not close to the source. However, the hop-based scheme also rejects requesters that are not within the hop limit (which are usually set to values such that users in the source’s neighborhood are given access), and therefore 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%.

For the malicious success rates we note that PDAC performs much better than the hop-based method for all cases except for the one mentioned above. We believe PDAC prevents malicious nodes from gaining access due to the use of experiential history in addition to the reports of black listed

users collected from the source’s immediate friends.

Table 2 compares the malicious success rates of PDAC when malicious node have different initial levels of notoriety. The results show that even at different levels of notoriety, PDAC still performs well by keeping the malicious node success rate fairly low compared to the hop-based method.

5.3 Security Analysis

Because the main goal of PDAC is to provide an access control mechanism for personal data, it is important for such a scheme to be resistant to malicious attacks. In this section, we consider few classes of 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 would 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 involves malicious attestors. We also show that PDAC effectively prevents both types of attacks. For clarity, we will assume in all cases that Alice is the owner of a data object to which Oscar, a malicious user, seeks access. Furthermore, we assume that by default Alice’s data object is inaccessible to Oscar, indicating that Oscar either falls in the deny zone or the attestation zone. If Oscar falls into the attestation zone, we assume that his request would eventually be rejected (i.e., Oscar won’t be able to acquire enough attester signatures). We denote the access limits set on the object by l and h with $0 < l \leq h$. The only way for Oscar to gain access to Alice’s object is to move to the accept zone for that object. This condition is captured in the following inequality:

$$d_{hop}(a, o) + d_{aff}(a, o) + d_{afdst}(a) + d_{pfdst}(a, o) \leq l$$

Since the last two terms $d_{afdst}(a)$ and $d_{pfdst}(a, o)$ in the above inequality are set by the owner (Alice), we will assume that both are set to zero to simplify the analysis. The term $d_{hop}(a, o)$ is fixed for Alice and Oscar, and using Equation (2) to substitute for $d_{aff}(a, o)$ the above inequality reduces to:

$$\lambda s_t + (1 - \lambda) \left(\frac{r_{oa} - a_{oa}}{q_{oa} + \Delta} \right) \leq l - d_{hop}(a, o)$$

Examining the previous inequality, we note that the only parameter which Oscar can actually manipulate is the long term success rate s_t because it is based on her request history. In contrast, the second term is based on the direct interaction between Oscar and Alice. Therefore Oscar could potentially collude with users in Alice’s social neighborhood in order to increase his long term success rate and gain access to the data object. Now, if we rearrange the last inequality and further use Equation (1) for s_t we get the following:

$$\left(\frac{r_t - a_t}{q_t}\right)\left(\frac{1}{1 + e^{-k/\alpha+\beta}}\right) \leq \frac{l - d_{hop}(a, o) - (1 - \lambda)\left(\frac{r_{oa} - a_{oa}}{q_{oa} + \Delta}\right)}{\lambda}$$

To simplify the above inequality we will denote the right hand side with Γ and replace the aggregate success rate $\frac{r_t - a_t}{q_t}$ with δ to get:

$$\delta\left(\frac{1}{1 + e^{-k/\alpha+\beta}}\right) \leq \Gamma \quad (4)$$

Because δ can take values only in $[-1,1]$, with value of $\Gamma < -1$ Oscar will not be able to gain access to Alice’s object. Therefore, Oscar will only have a chance if the value of Γ is in $[-1,0)$. For example if the object has a lower limit $l = 0.8$ and if we assume that $d_{hop}(a, o) = 1$, $\lambda = 0.25$, and the success rate of Oscar for Alice’s data is 0.1 we get $\Gamma = -1.1$. If the success rate is 0 (i.e., Alice didn’t have any previous interaction with Oscar) then the value of Γ would be -0.8 . We also note that Γ can only be negative in this case because a nonnegative value would mean that Oscar already satisfies the requirement and has access to the object which we assumed he doesn’t. Because the scaling factor depends on the number of unique data posters in the access history, Oscar needs to find enough users to collude with so that the scaling factor would not increase the aggregate success rate needed to get access to the data object.

In the second type of attack we assume that Oscar falls into the attest zone and we also assume the existence of malicious users among the attesters. If Oscar is able to collude with the malicious attesters and if the number of bad attesters is enough to meet the requirements set by Alice (the minimum number of signatures required from the attesters), then Oscar can acquire enough illegitimate signatures to access the data object. A possible solution to prevent such an attack is to require the majority of the attesters to sign on the RFA before access is granted. The majority of attesters solution is effective especially in the context of social networks because data owners usually choose as attesters their friends or best friends i.e. users who are highly trusted and therefore it is unlikely that the majority of the attesters will be malicious.

6. PDAC IMPLEMENTATION

We have implemented a working prototype of the PDAC on top of a social network emulator called the *Social Utility Networking Environment* (SUNE). The SUNE infrastructure and PDAC modules were developed in Ruby. The 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 attester nodes in determining whether a requesting node should be given access to a particular data item.

The prototype currently allows users to upload data items onto the Content Manager and request access to uploaded data items. Posting a new data item onto the Content Manager requires the user to define the accept, attest, and deny zones, choose a strict or relaxed data propagation level, assign peers as attesters for the data object, and specify the number of attester signatures required for peers that fall in the attest zone. The affine distance is calculated by the Trust Manager using interactions from the requester’s social neighborhood. 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. For automatically attested data objects, each attester uses the hop distance criteria specified by the publisher.

To implement the Data Leak Manager, an open source application called DuMP3 (<http://dump3.sourceforge.net>) was used, which uses a variety of different fingerprinting algorithms to determine file similarity. The *strict* and *relaxed* privacy settings have also been implemented, and the confidentiality settings for a reposted item is adjusted according to the applicable privacy setting.

The current prototype also includes a blacklist feature, which allows the user to immediately prevent certain peers from gaining access to her data items regardless of their trusted distance. A user’s blacklist is shared among her immediate peers, but each user can override the shared blacklist with her own local one.

In a future version, we intend to include interactions with like-minded peers in the trust distance computation.

		Malicious success rate	
Request dist.	Outcome dist.	50% Initial notoriety	10% Initial notoriety
discrete shallow	discrete steep	0.013	0.080
discrete shallow	discrete shallow	0.031	0.158
random	discrete shallow	0.011	0.027

Table 2: Comparison of malicious success rates for 50% initial notoriety and 10% initial notoriety.

7. RELATED WORK

The *discretionary access control* (DAC) [16] has been one of the cornerstones of computer systems protection since 1970s. In DAC, the owners of data objects have the exclusive privilege of deciding who gets what type of access to their data objects. One of the major issues with this “owner-centric” DAC is the propagation of access rights. The literature [5] on DAC has many approaches for addressing the access propagation problem using ideas such as copy flags and grant options. Like DAC, PDAC allows owners to express their discretion on the sharing process. In particular, the owner can setup the trust zones by specifying the trusted distance values. However, user-to-zone placements are done collaboratively exploiting structural properties of the underlying social network.

The *mandatory access control* (MAC) [4] is another protection scheme proposed first in the 1970s that aims to enforce lattice-based information flow constraints to create high-assurance information systems with confidentiality, integrity, and aggregation concerns. In practice, *covert channels* remain a major impediment for implementing MAC. As a result, MAC-based schemes have found limited applicability in highly legislated environments such as the military. Because the PDAC deals with personal data it implements confidentiality constraints.

Role-based access control (RBAC) [13] has in the past decade emerged as the most widely discussed alternative to DAC and MAC. The basic idea of RBAC is to create well-defined abstractions called roles and assign permissions to users via the roles. Due to strong commercial interest, RBAC has seen lot of research and development. However, enterprises should perform careful security engineering analysis to determine the best possible deployment of RBAC. This manual administrative intervention is identified as a key future research direction by Bertino et. al. in [6]. With the large body of work centered around RBAC, below we discuss selected few that use trust to create agile RBAC frameworks.

An XML-Based RBAC (X-RBAC) framework that incorporates context-based access control for dynamic XML-based web services is provided in [8]. This is enhanced further in [7] to make the role-to-user assignments trust based. In [7] the trust is interpreted as the level of confidence associated with a user based on certain certified attributes. The certification can be provided by trusted third parties (e.g., a PKI certification authority). The system can subsequently adjust the trust levels based on the user's activity profile that is maintained by the system. The lack of scalability of traditional mechanisms is one of the primary motivations behind X-RBAC. X-GTRBAC is a framework based on the Generalized Temporal Role Based Access Control (GTRBAC) [14] and provides a generalized mechanism to express a wide range of temporal constraints including periodic and duration constraints on roles, user-role assignments, and role-permission assignments. The X-GTRBAC augments GTRBAC with XML to allow policy enforcement in heterogeneous and distributed environment.

Another trust-based extension of RBAC called *TrustBAC* is presented in [10]. The TrustBAC assigns trust levels to users based on three factors (experience, knowledge, and recommendation). The experience component is based on the user's past behavior. The knowledge component (knowledge about other characteristics of the user for example credentials) has two parts - direct knowledge and indirect knowledge the weighted sum of the two parts constitutes the knowledge component. The recommendation component is evaluated based on recommendation values provided by others about a user, the recommendation values are weighted by the reputation of the recommender. The system computes overall trust levels based on these parameters and are assigned to roles.

In certain ways, the PDAC is similar to RBAC and its trust-based extensions. Like the roles in RBAC, the PDAC introduces zones, which are defined by the data originators and other users are mapped onto the different zones by the community. Using zones, a data originator can set propagation limits on his/her data objects.

In [9], Peregó 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 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 much user a considers user b trustworthy 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 the model each owner states the access control policies according to his/her preferences. While each requester is in charge of verifying to the owner whether he/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. The requesting node provides the owner with a proof showing that it actually satisfies the policies. Proofs are generated using a reasoner therefore allowing 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. To deal with the large number of certificates special Central Nodes are introduced to manage the certificates and deal with requests for different certificates made by the users.

In certain aspects, The rule-based access control model is similar to PDAC in that both schemes exploit the structural properties of the social network in the access control model. In PDAC the social based hop distance between the users in addition to the activities occurring in social neighborhoods are utilized in mapping the users into zones. While in the rule-based scheme the type and depth or relationships of the social network are used to express the access control policies. However the zone mapping in PDAC allows it to create a dynamic overlay on top of the static social network where the activities of users on the social network allows for an adaptive overlay, whereas the rule-based scheme relies on a static social network. Finally, PDAC considers an important aspect of access control which is data leaks – it prevents the users from using the social network to redistribute protected data. The rule-based scheme does not consider this issue.

In addition to the trust-based extensions of RBAC, there is significant effort in developing “trust-based” access control mechanisms from scratch [1, 22, 12]. In [22], a trust-based access control framework for peer-to-peer (P2P) file sharing is presented. 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 behavior and contribution models the cooperative behavior. The trust and contribution scores are computed using direct and indirect measurement values.

In [1], a trust based access control system for mobile ad-hoc collaborative environment is provided. The system com-

bines 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.

In [12], 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.

8. CONCLUSIONS AND FUTURE WORK

In this paper, we presented PDAC, a protection scheme for personal data stored online. The PDAC leverages the social structure present in online social networks. However, it can be applied to control accesses to personal data both on and off social networks.

Central to the PDAC scheme is a trusted distance measure that is computed based on social network structural information and experiential data of the users. Using the trusted distance, a data object owner defines three protection zones (like roles in RBAC). The PDAC uses a collaborative computing approach to map other users with respect to the data protection zones defined by the owner of a data object. Based on which zone a user is mapped into, her requests to access the data objects of another user will be accepted, further processed, or rejected by PDAC. The requests further processed by the PDAC will undergo another round of evaluation by a set of attestors designated by the owner of the data object.

Unlike many "discretionary" access systems, PDAC goes beyond the first access. It looks at subsequent postings to determine potential data leakages. The idea is to comprehensively enforce the confidentiality requirements of the data object owner.

We performed simulations using data from the `myspace.com` social network to compare the performance of PDAC with a hop-based scheme. We found that PDAC has a higher success rate and is more efficient in screening requests made by malicious users. This demonstrates the flexibility of PDAC compared to the hop-based scheme.

Currently, we have a working prototype of the PDAC scheme implemented on top of a social network emulator. Future work will investigate incorporating the opinions of like-minded individuals in the calculation of affine distances.

9. REFERENCES

- [1] ADAMS, W., AND DAVIS, N. Toward a decentralized trust-based access control system for dynamic collaboration. In *Proc. of the 2005 IEEE Workshop on Information Assurance and Security* (June. 2005).
- [2] AGGARWAL, G., BAWA, M., GANESAN, P., GARCIA-MOLINA, H., KENTHAPADI, K., MISHRA, N., MOTWANI, R., SRIVASTAVA, U., THOMAS, D., WIDOM, J., AND XU, Y. Vision paper: Enabling privacy for the paranoids. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases* (Aug. 2004), pp. 708–719.
- [3] BARTAL, Y., MAYER, A., NISSIM, K., AND WOOL, A. Firmato: A novel firewall management toolkit. *ACM Transactions on Computer Systems* 22, 4 (Nov. 2004), 381–420.
- [4] BELL, D. E., AND LAPADULA, L. J. Secure Computer Systems: Mathematical Foundations. Tech. Rep. MTR-2547, The MITRE Corporation, Bedford, MA, Mar. 1973.
- [5] BENANTAR, M. *Access Control Systems: Security, Identity Management and Trust Models*. Springer, New York, NY, 2006.
- [6] BERTINO, E., KHAN, L. R., SANDHU, R., AND THURASINGHAM, B. Secure knowledge management: Confidentiality, trust, and privacy. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 36, 3 (May 2006).
- [7] BHATTI, R., BERTINO, E., AND GHAFOR, A. A trust-based context-aware access control model for web-services. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services* (June 2004), pp. 184–191.
- [8] BHATTI, R., JOSHI, J., BERTINO, E., AND GHAFOR, A. Access control in dynamic xml-based web-services with x-rbac. In *ICWS '03: Proceedings of the 1st International Conference on Web Services* (June 2003), pp. 243–249.
- [9] CARMINATI, B., FERRARI, E., AND PEREGO, A. Rule-based access control for social networks. In *proceedings of OTM Workshops (2)* (2006), pp. 1734–1744.
- [10] CHAKRABORTY, S., AND RAY, I. 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, 2006), pp. 49–58.
- [11] DE AGHION, B. A., AND MORDUCH, J. *The Economics of Microfinance*. MIT Press, 2005.
- [12] DIMMOCK, N., BELOKOSZTOLSKZI, A., EYERS, D., BACON, J., AND MOODY, K. Using trust and risk in role-based access control policies. In *Proceedings of the ninth ACM symposium on Access control models and technologies* (June. 2004).
- [13] FERRAILOLO, D., AND KUHN, R. Role-based access controls. In *Proceedings of the 15th NIST-NCSC National Computer Security Conference* (Oct. 1992), pp. 554–563.
- [14] JOSHI, J., BERTINO, E., LATIF, U., AND GHAFOR, A. A generalized temporal role-based access control model. *IEEE Transactions on Knowledge and Data Engineering* 17, 1 (Jan. 2005), 4–23.
- [15] KARLAN, D. Social connections and group banking. *Economic Journal* 117 (Feb. 2007), F52–F84.
- [16] LAMPSON, B. W. Protection. In *Proc. 5th Princeton Symp. Information Science and Systems* (Mar. 1971), pp. 437–443. Reprinted in *Operating Systems Rev.* 8, 1 (Jan. 1974), 18–24.
- [17] MORDUCH, J. The microfinance promise. *Journal of Economic Literature* 37, 4 (1999), 1569–1614.
- [18] PRAKASH, V. V., AND O'DONNELL, A. Fighting spam with reputation systems. *ACM Queue* 3, 9 (2005), 36–41.

- [19] RAMAKRISHNAN, R., AND TOMKINS, A. Towards a peopleweb. *IEEE Computer* 40, 8 (Aug. 2007), 63–72.
- [20] SCHLEIMER, S., WILKERSON, D. S., AND AIKEN, A. Winnowing: local algorithms for document fingerprinting. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data* (June 2003), pp. 76–85.
- [21] SHIVAKUMAR, N., AND GARCIA-MOLINA, H. Building a scalable and accurate copy detection mechanism. In *Proceedings of the 1st ACM International Conference on Digital Libraries* (Mar. 1996), pp. 160–168.
- [22] TRAN, H., HITCHENS, M., VARADHARAJAN, V., AND WATTERS, P. A trust based access control framework for p2p file-sharing systems. In *Proc. of the 38th Annual Hawaii International Conference on System Sciences* (Jan. 2005).
- [23] VILLEGAS, W. Personal data protection on online storage systems. Master's thesis, School of Computer Science, McGill University, Montreal, 2008 (under preparation).
- [24] WASSERMAN, S., AND FAUST, S. *Social Network Analysis: Methods and Applications*. Cambridge University Press, Cambridge, UK, 1992.
- [25] WOOL, A. A quantitative study of firewall configuration errors. *IEEE Computer* 37, 6 (2004), 62–67.