Enabling Interoperable and Selective Data Sharing among Social Networking Sites

Dongwan Shin and Rodrigo Lopes

Computer Science Department,
New Mexico Tech,
Socorro, NM 87801, USA
{doshin,rodrigo}@nmt.edu
http://sislab.cs.nmt.edu

Abstract. With the widespread use of social networking (SN) sites and even introduction of a social component in non-social oriented services, there is a growing concern over user privacy in general, how to handle and share user profiles across SN sites in particular. Although there have been several proprietary or open source-based approaches to unifying the creation of third party applications, the availability and retrieval of user profile information are still limited to the site where the third party application is run, mostly devoid of the support for data interoperability. In this paper we propose an approach to enabling interopearable and selective data sharing among SN sites. To support selective data sharing, we discuss an authenticated dictionary (ADT)-based credential which enables a user to share only a subset of her information certified by external SN sites with applications running on an SN site. For interoperable data sharing, we propose an extension to the OpenSocial API so that it can provide an open source-based framework for allowing the ADT-based credential to be used seamlessly among different SN sites.

Keywords: Privacy, social networking, selective data sharing, interoperability, authenticated dictionary.

1 Introduction

Online social networking (SN) sites have emerged in the last few years as one of the primary sources for information exchange among friends and acquaintances. Typical examples of SN sites are networks of individuals (people in a specific culture, in communities, or in similar working contexts) or networks of organizations (companies, cities, political parties) [12]. Most of them offer the basic features of online interaction, communication, and sharing affinities, by allowing individuals to create digital profiles, which will be then viewed by others. Hence, personal information works as a fundamental building block for the proper operation in online SN sites. As the use of personal information in social networks becomes manifold, including the representation of an individual's identification, so does the abuse or misuse of the information. One of the most important issues

E. Bertino and J.B.D. Joshi (Eds.): CollaborateCom 2008, LNICST 10, pp. 439–450, 2009.

to be immediately addressed in this context is the issue of privacy of sensitive personal information, which is generally any type of data which could be used to cause significant harm to the individual.

In addition to that, as the number of different social services provided in a diverse context grows, an individual tends to possess many different profiles across services, each of which contains a range of information that may be duplicate, thus making an issue of data interoperability in SN sites of great importance. This is especially true for the use of third-party applications within SN sites; a new kind of usage of SN sites has developed with the development of small applications that run inside the user profile in a given SN site, many times making use of his/her profile information, thus extending the functionality of the original SN site. Although there have been several proprietary or open source-based approaches to unifying the creation of third party applications, the availability and retrieval of user profile information are still limited to the SN site where the third party application is run, mostly devoid of the support for data interoperability.

In this paper we propose an approach to enabling interopearable and selective data sharing among SN sites. To support selective data sharing for better privacy, we discuss an authenticated dictionary (ADT)-based credential which enables a user to share only a subset of her information certified by external SN sites with applications running on an SN site. Specifically, our approach allows a user to obtain credentials from an SN sites and be able to show that credential for data sharing without the need to contact the SN site again. More importantly, our approach allows the credential to be constructed efficiently using a cryptographic hash function and the user to show and prove any subset of her attributes in the credential. For interoperable data sharing, we propose an extension to the OpenSocial API so that it can provide an open source-based framework for allowing the ADT-based credential to be used seamlessly among different online SN sites. The main contributions of this paper are 1) a novel application of ADT into a social networking domain, 2) the development of a credential based system, not relying on expensive cryptographic constructs, and 3) the extension of the OpenSocial API for data interoperabiliity among SN sites.

The rest of this paper is organized as follows. In Section 2, we discuss background material related to the privacy issues in SN sites and the OpenSocial initiative. Section 3 discusses our approach, followed by the discussion of current implementation based on the proposed approach in Section 4. Section 5 discusses our future work and Section 6 concludes this paper.

2 Background and Related Technologies

2.1 Online Social Networks and Privacy Issues

Online social networks have rapidly emerged, diversified, and adopted in recent years, and the wide adoption of the Internet contributed to the recent thriving popularity of those social networking sites. In general, the privacy issue in online social networking is closely related to the level of identifiability of the information provided, its possible recipients, and its possible uses [12]. The identifiability of

information is quite challenging in a sense that even online social networking sites that do not expose their users' identities may provide enough information to identify the profile's owner. According to a recent study [13], for instance, a 15 percent overlap of personal information is observed in two of the major social networking sites, which allows sophisticated viewers to know more about an individual than she may want them to. In addition, since individuals often re-use the same or similar photos across different sites, an identified face can be used to identify a pseudonym profile with the same or similar face on another site. The possible recipients of personally identifiable/identified information are social network hosting sites and third party application hosting sites that may abuse or misuse the information. Our current approach is related to this aspect in that it is based on the notion of user centricity in sharing user profile in a sense that the user will have the ultimate authority to share which data in her profile with what SN sites, thus providing the user more control on her profile.

2.2 OpenSocial and Profile Sharing

The OpenSocial project [3] was born to provide application developers with a single API that would allow the development of applications that would run across different websites, addressing the issue of a different social application API for each SN site. The OpenSocial API, implemented by each provider, would allow a developer to build an application once and deploy it on different websites without any modification to the code. On the other hand, it would also allow upcoming SN sites to have support to a diverse portfolio of applications already developed for popular and established SN websites, and at the same time saving the time and effort of designing and building their own social application API.

The API is divided in three main areas according to the functionality provided. These areas are People and Relationships, Activities and Persistence. People and relationships provide functionality to access users' information and the relations that users have with each other. The activities functionality allows the application developer to access the users' activities in the context of the website and finally the persistence API provides functionality for the application to store and retrieve application specific information. The API views users from three different perspectives: viewer, owner and friends. The viewer is the person in whose browser the application is being rendered and displayed while the owner is the person who owns the profile or information being displayed. Friends refer to the social connections of either the viewer or the owner. The owner and viewer may be the same person if, for example, someone is looking into his own profile and an application is running on that profile, using the owners information who happens to be also the one person looking at the displayed web page.

One of the main limitations in the OpenSocial is the inability for an application to access user information from different websites. Recently, different online SN sites have proposed mechanisms to allow the user profile to be exported to other websites, social networking or not. Some of these initiatives are MySpace Data Availability and Facebook Connect [2,1]. However, the sharing of information on these initiatives is one way, in other words, the information can be

shared from the SN website with other websites, but not vice-versa. Also, the source SN website is directly involved with the sharing of user profiles, gaining the knowledge of all the services the user is sharing his information with, and thus having an implication of privacy violation. Our approach put forward in the paper addresses these issues seamlessly through the use of a credential system.

2.3 Credential Systems

A credential system is a system in which a user can obtain credentials (i.e., signed statements) from one organization and demonstrate possession of them to other organizations, and several credential systems based on number-theoretic schemes have been proposed for different purposes in literature. Chaum's approach to designing the digital cash system [9,10], based on blind signature techniques, was one of them, also called an anonymous credential system. This system made it possible to obtain a certified statement from an issuer and show it to a verifier without the possibility of tracing the use of credentials. The credential was built upon the number theory, especially the use of an exponent that defines the type and the value of the credential in the blind signature. One major disadvantage of using this system is that a trusted third party is always required that all participating entities are dependent upon.

Similar to Chaum's system, but a more advanced scheme to design an anonymous credential system was presented by Brands [6]. His credential system could support many features such as expressions of any satisfiable proposition from proposition logic, limitation on the number of times a credential may be used, revocable anonymity, and discouragement of lending credentials. Camenisch et al. [8,7] proposed a credential system that relies heavily on proofs of knowledge like Brands' system. One of the main disadvantages in these credential systems is obviously related to the expensive computational aspect of their cryptographic primitives using number theory and zero-knowledge proof (ZKP).

3 Our Approach

An interoperable and selective data sharing approach is discussed in this section. We first discuss the concept of ADT, then discuss how to construct attribute credentials based on that, and finally present how to integrate the ADT-based credential into OpenSocial-based social networking sites.

3.1 Selective Disclosure Credentials

Authenticated dictionaries (ADTs) have been primarily studied and used in the context of certificate revocation in public key infrastructure (PKI), especially to implement certificate revocation lists (CRLs). One of the best known examples is based on the Merkle hash tree in [15]. We decided to use an authenticated dictionary based on skip lists [5,11] for our purpose of building a credential system that allows the user to selectively share their attributes [14]. A skip list,

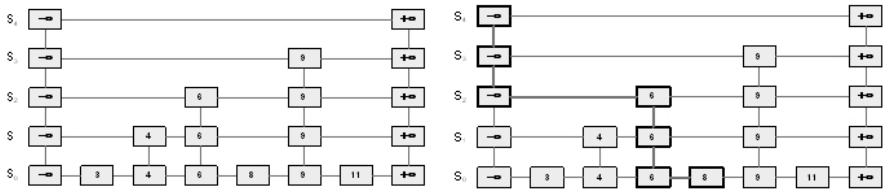

Fig. 1. Skip List (left) and A Value Search in Skip List (right)

shown in Figure 1, is a data structure that allows the effective search and update of elements within a set. It supports the following operations on a set of elements: find(x), insert(x), and delete(x), where x is an element of the set.

Initially, all elements will be ordered and form the first list from the bottom. Subsequent lists are built randomly by selecting elements from the previous list. The elements will be selected with a probability of $\frac{1}{2}$. This will be repeated until there is only one element. The first and last symbols ($-\infty$ and $+\infty$) represent the lower and higher possible boundaries, the last list consisting of only these two symbols. The search of an element in the list is started at the lower boundary symbol on the top list and continues to the right until we find the element being searched or an higher element. If the element is lower, we will descend to the element immediately below; else we will descend to the element below the previous symbol in the list. The search ends when we find our target at the bottom list or two consecutive elements at the bottom list in which the first is lower than our target, and the second is larger. The latter proves that the element being searched is not in the list. The process for searching the value 8 is shown in Figure 1. The data structure described above is very efficient for search, whose cost is $\mathcal{O}(\log n)$.

A special function f is necessary to introduce the *authenticated* skip list. The function is a commutative hashing function, that is, a hashing function that takes two values and returns the same hash independent of the order the values are given. The function is calculated for each element and will depend on all previous elements, as shown in Figure 2. The lower boundary element on the top list will contain a tag as a result of the function on the element that actually depends

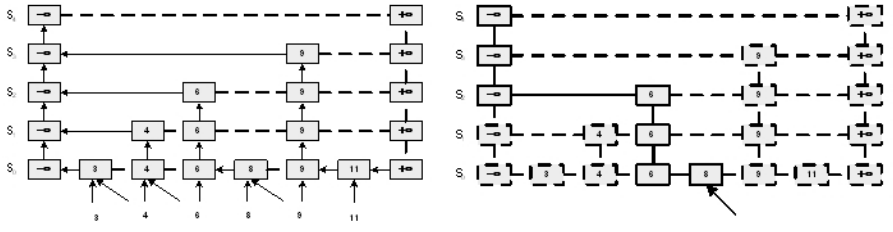

Fig. 2. Commutative hash computation flow (left) and values needed to authenticate the result of search (right)

on the full list. This last tag will be signed and therefore be used to verify the authenticity of the skip list or dictionary. [5,11] has more details on this. The commutative hashing is used to allow the function to be computed independently of the order in which both parameters of the function are entered, making the verification process more efficient, and the right part of Figure 2 shows the values needed to authenticate the result of a search.

The structure of ADT can be used to represent a credential and allow the user to disclose a subset of her attributes to verifiers. The values inside each of the nodes of ADT will be replaced by hashes of user attributes. ADT-based credentials will not contain any user information, either in plain text or encrypted, but only hash values of attributes. Hence, the basic security of this credential system resides in the selection of a good cryptographic hash function. Since the elements in the dictionary are stored in order of the value of their hash, there is no predefined order in which particular attribute values are stored. The issuer will then sign the dictionary.

There are two different ways an attribute can be proved to be an element of ADT. In the first method, the user sends the verifier a set of attributes she wants to disclose, the values in the path from an attribute in the set necessary to compute the f function value for the last element (the lower boundary element in the top list), starting at the element whose presence is to be proved; and finally the signed f function value. Verifier will then hash the attribute and use the values sent from the user to recompute the f function value of the last element, which will in turn be verified to be legitimate with the signature. Note that alternatively, after the user releases the set of her attributes to be disclosed, the verifier can interactively request the values in the path and the signed f function value to the user for each attribute in the set to be proved. A second method uses the full dictionary instead of using the values in a path for proof. The user sends ADT, a set of attributes she wants to disclose, and the signed f function value. The verifier computes function f for each node in the ADT. After verifying the correctness of the f function value of the last element and its signature, the verifier hashes each one of the attributes and searches its value in ADT. If found, the attribute is a legitimate member of the dictionary.

3.2 Extensions to the OpenSocial API

The OpenSocial API allows a third-party application to use a user's profile data available from the container host SN sites. The extension proposed in our approach would allow the application to request information about the user from different websites. To allow this, the DataRequest object of the OpenSocial API needs to be modified, as shown in Figure 3.

The DataRequest object was subclassed by two objects called DataRequestfromRemoteSite and DataRequestfromCredential respectively. Therefore, the data could be retrieved and shared from the container host website, a remote website, and directly from the user through the selective disclosure credential presented in Section 3.1. From the perspective of the application developer, the data from different sources will have the same format as data retrieved from the container

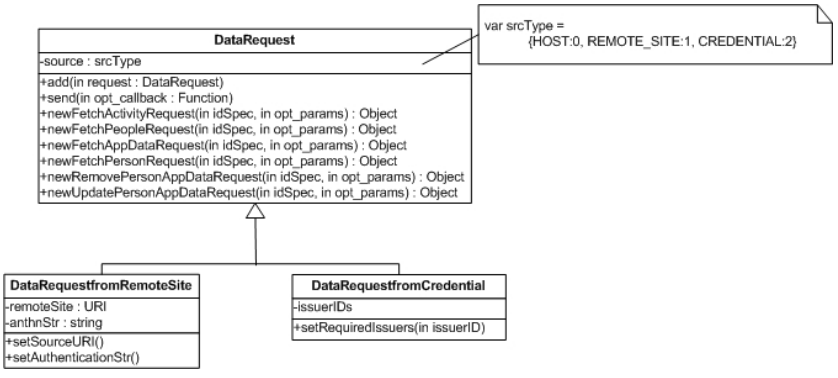

Fig. 3. UML representation of DataRequest, DataRequestfromCredential, DataRequestfromRemoteSite objects and their relationship

host site. The major difference between the two external data source options is the involvement of the SN website that hosts the data. If the data source is an URI, the host SN website is assumed to know what user information can be disclosed and to whom on the basis of user preference set on the site. If the data source is a credential, the SN site having issued the credential has no knowledge of what information is being shared and with whom. A third-party application may have requirements to which SN sites it supports, and the issuerIDs attribute refers to the SN sites that issue credentials. This field is optional. If set, only credentials issued by the given issuer or set of issuers can be used. If not set, any credential that provides the requested information can be used.

In order to create new data request to not just the container but also the remote site and the credential, the opensocial object also needs to be modified. A new method called newDataRequestEx is added to the object for that purpose, as shown in Figure 4. Depending on the parameter that specifies the source of data, the method will create an appropriate data request object as discussed previously.

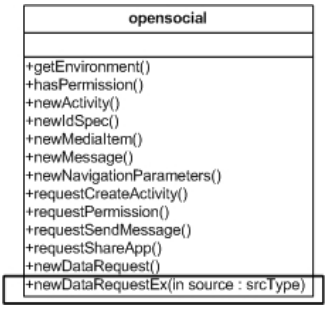

Fig. 4. UML representation of opensocial object and the addition of new method called newDataRequestEx

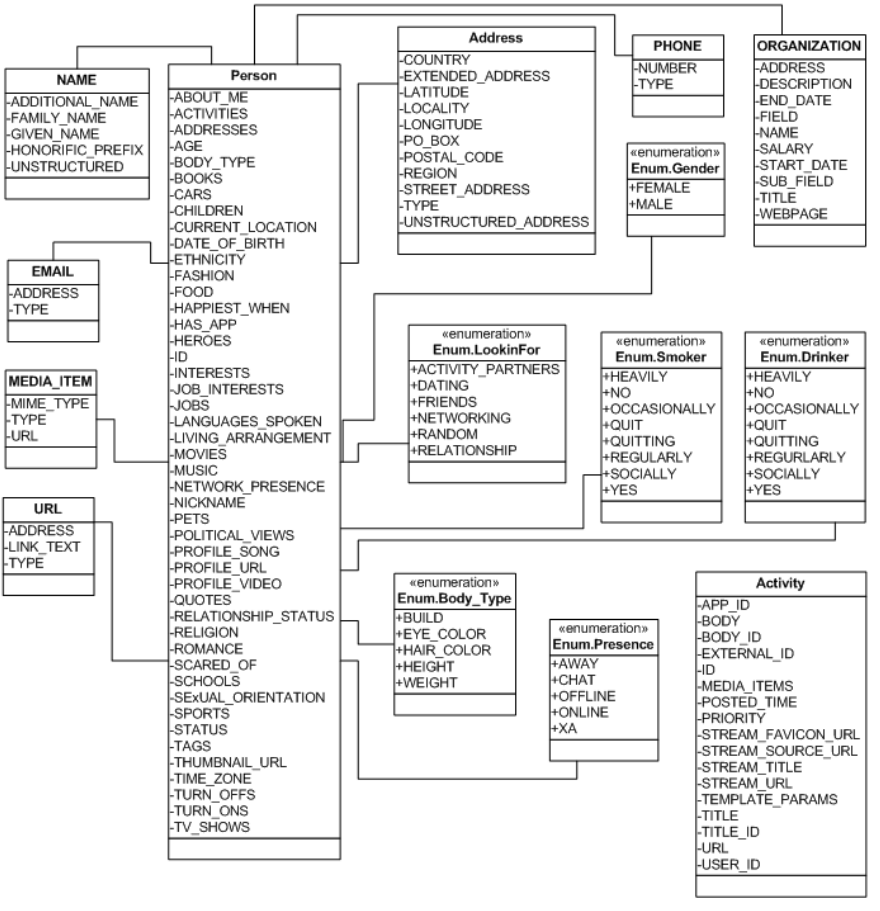

Fig. 5. Representation of the objects as described in the OpenSocial API [4]

3.3 Attribute Mapping from OpenSocial to ADT-Based Credential

All the information that can be retrieved by an OpenSocial-based third party application must be supported as an attribute of the selective disclosure credential. This means that a mapping must exist that allows SN sites to both build and read the credential. The information that can be retrieved from an SN site is limited to two top level objects and the relationship between them. These two top level objects are Person and Activity [3]. Every other object is a part of the top level objects. A more detailed diagram can be seen in Figure 5.

The basic structure of an attribute inside the ADT-based credential is as follows:

[attribute type]:[attribute ownership]:[attribute value]

The attribute type is no more than the type as defined in the OpenSocial API. For the attribute value there can be more than one meaning. If the type is an object, the value will be a set of references to the actual types that constitute it, every object ultimately leading to literal types. If the type is a literal, then the value is the actual literal representation. As previously defined in Section 3.1, attributes in the credential are hashes of the actual attribute values, therefore the value for objects will be sequences of hash values for the actual components of that object. Those components are also part of the credential and can be found in the same credential, if the user chooses to disclose them. The attribute ownership will only have a meaning for top level objects, as it will represent the relationship between the owner of the credential and the owner of the information, that is, the object represented in the attribute. They can be the same user, in which case the information belongs to the owner of the credential, or they can be different users, meaning the owner of the credential has some form of relation to the owner of the information. The OpenSocial API does not specify a limit to how deep in the social graph can information be retreived [4] , it is left up to the container site to define how deep can a user retrieve friends information.

4 Current Implementation

Our implementation phases were divided into two: the first was to develop stand-alone prototypes based on functionalities such as ADT-based credential issuance, management, and verification. The second was to port these prototypes into the OpenSocial-based web environment. As of writing this paper, we finished the first phase of our implementation.

As discussed in previous section, attributes are hashed and then inserted into ADT. The attribute is strongly typed and represented as a concatenation of the attribute type and attribute value, separated by a colon[1]. As a container for the attribute in ADT, we decided to use a four-pointer node. The node will contain pointers to all its neighbors: Up, Down, Left and Right. In addition to the attribute, the f function value will be stored in the node as well. An attribute can be proved to be an element of ADT by finding it inside the skip list and retrieving the path function values. The user is responsible for doing this. The user will send only the attribute's type and value and the corresponding values in the path that allow the re-computation of the signature element. An interesting observation is that the user will most likely disclose several attributes, and these attributes are likely to have overlapping path values. In such cases, the user only needs to send each repeating value once. To allow the user to prove ownership of the credentials there is the need to communicate information to the verifier. This is done through an XML file that includes all different values that will be used, either hashes of attributes or function values and a list of shared attributes,each one including the attribute type, attribute value and list of values needed to calculate the signed value.

[1] Note that our current implementation includes only the attribute type and value.

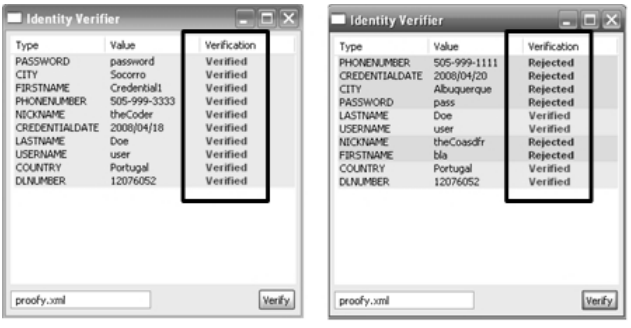

Fig. 6. The User Interface for Verifying ADT-based Credentials

Fig. 7. The User Interface for Managing ADT-based Credentials

Revocation is an important issue in credential-based systems. One intuitive way to support revocation in our design is to have the SN to ask the UAP if any given credential is revoked or not, as typical in current PKI systems. Additionally, the UAP has validity periods associated to credentials it issues.

The standard widget toolkit (SWT) was used for developing the prototype UI for issuing, using, and verifying ADT-based credentials. The credential issuance module was implemented that allows the construction and issuance of credentials. The credential verification module was also implemented to test the functionality required for the verification purpose. This module reads the XML proof request generated by the credential management module and verifies all the attributes disclosed, showing the ones that were correctly verified and those that failed verification, as shown in Figure 6. The credential management module,

depicted in Figure 7 includes a card picker and an attribute selector. The list of attributes displayed on the attribute selector matches the selected card on the card picker. In addition, it also includes a graphical representation of ADT that shows the full dictionary. Finally, it includes a simple proof request viewer/editor that shows the request built for the currently selected attributes. The editor allows that request to be manually changed, providing a valuable tool to build invalid requests in an attempt to fool the verifier and test our current prototype system.

5 Future Work

We are currently working on the second phase of our implementation using Apache Shindig and its extension, SUN SocialSite. The open source project Apache Shindig provides a platform that allows us to port our prototypes into a web environment. Shindig has been used successfully as a base for the OpenSocial containers such as Hi5 and Orkut. It implements not only the OpenSocial API but also the gadget engine. The current implementation of Shindig is being modified to accommodate our extension. More specifically, we are working on the changes in the object structures and JavaScript functionality that third party applications can use based on our extension. The server side implementation of Shindig will also be modified. Since Shindig has both Java and php implementations, we are going to focus only on Java version, reusing all the Java code that has already been developed to implement our prototypes in the first phase.

6 Conclusion

With the rapid growth of online social network (SN) sites, there is an increasing demand for sharing user profiles among those sites in a secure and privacy-preserving manner. In this paper we discussed a novel approach to developing an interopearable and selective data sharing solution among such SN sites. Our approach was mainly based on an authenticated dictionary (ADT)-based credential which allows a user to share only a subset of her information certified by external SN sites with applications running on an SN site. Additionally an extension to the OpenSocial API for the purpose of supporting data interoperability was discussed. Finally we presented a proof-of-concept implementation based on our approach, along with our future work.

Acknowledgement

This work was partially supported at the Secure Computing Laboratory at New Mexico Tech by the grants from Sandia National Laboratories (PASP10) and from New Mexico Tech VP Office.

References

1. Facebook Connect, http://developers.facebook.com/connect.php
2. MySpace Data Availability (DA), http://developer.myspace.com
3. OpenSocial Foundation, http://www.opensocial.org
4. OpenSocial pages at Google Code, http://code.google.com/apis/opensocial
5. Anagnostopoulos, A., Goodrich, M.T., Tamassia, R.: Persistent authenticated dictionaries and their applications. In: Proceedings of 4th International Conference on Information Security, Malaga, Spain, October 1-3 (2001)
6. Brands, S.: Rethinking Public Key Infrastructure and Digital Certificates - Building in Privacy. MIT Press, Cambridge (2000)
7. Camenisch, J., Herreweghen, E.V.: Design and implementation of the idemix anonymous credential system. In: Proceedings of 9th ACM Conference on Computer and Communication Security, Alexandria, VA, November 7-11 (2002)
8. Camenisch, J., Lysyanskaya, A.: A signature scheme with efficient protocols. In: Proceedings of 3rd Conference on Security in Communication Networks, Amalfi, Italy, September 12-13 (2002)
9. Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. Communications of the ACM 28(10), 1030–1044 (1985)
10. Chaum, D.: Achieving electronic privacy. Scientific American, 96–101 (August 1992)
11. Goodrich, M.T., Tamassia, R., Schwerin, A.: Implementation of an authenticated dictionary with skip lists and commutative hashing. In: DISCEX II (2001)
12. Gross, R., Acquisti, A., Heinz III., H.J.: Information revelation and privacy in online social networks. In: Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society, Alexandria, VA, November 7 (2005)
13. Liu, H., Maes, P.: Interestmap: Harvesting social network profiles for recommendations. In: Proceedings of IUI Beyond Personalization 2005: A Workshop on the Next Stage of Recommender Systems Research, San Diego, CA, January 9 (2005)
14. Lopes, R., Shin, D.: Controlled sharing of identity attributes for better privacy. In: Proceedings of the 2nd International Workshop on Trusted Collaboration, White Plains, USA, November 12-15 (2007)
15. Merkle, R.C.: A digital signature based on a conventional encryption function. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, pp. 369–378. Springer, Heidelberg (1988)