A Public-Key Protocol for Social Networks with Private Relationships

Josep Domingo-Ferrer

Rovira i Virgili University
UNESCO Chair in Data Privacy
Department of Computer Engineering and Mathematics
Av. Països Catalans 26, E-43007 Tarragona, Catalonia
josep.domingo@urv.cat

Abstract. The need for protecting the privacy of relationships in social networks has recently been stressed in the literature. Conventional protection mechanisms in those networks deal with the protection of resources and data, *i.e.* with deciding whether access to resources and data held by a user (owner) should be granted to a requesting user (requestor). However, the relationships between users are also sensitive and need protection: knowing who is trusted by a user and to what extent leaks a lot of confidential information about that user. The use of symmetric key cryptography to implement private relationships in social networks has recently been proposed. We show in this paper how to use public-key cryptography to reduce the overhead caused by private relationships.

Keywords: Social networks, Privacy, Private relationships, Public-key cryptography.

1 Introduction

Social networks have become an important web service [7] with a broad range of applications: collaborative work, collaborative service rating, resource sharing, searching new friends, etc. They have become an object of study both in computer and social sciences, with even dedicated journals and conferences. They can be defined as a community of web users where each network user can publish and share information and services (personal data, blogs and, in general, resources). In some social networks, users can specify how much they trust other users, by assigning them a trust level [1,6]. It is also possible to establish several types of relationships among users (for example, "colleague of", "friend of ", etc.). The trust level and the type of relationship are used to decide whether access is granted to resources and services being offered.

As pointed out in [4], the availability of information on relationships (trust level, relationship type) has increased with the advent of the Semantic Web and raises privacy concerns: knowing who is trusted by a user and to what extent discloses a lot about that user's thoughts and feelings. See [2] for an analysis of related abuses.

These privacy issues have motivated some social networks [5,8] to enforce simple protection mechanisms, according to which users can decide whether their

V. Torra, Y. Narukawa, and Y. Yoshida (Eds.): MDAI 2007, LNAI 4617, pp. 373–379, 2007.

resources and relationships should be public or restricted to themselves and those users with whom they have a direct relationship. Unfortunately, such straightforward mechanisms results in too restrictive policies.

In [3], a more flexible access control scheme is described, whereby users can be authorized to access a resource even if they have no direct relationship with the resource owner, but are within a specified depth in the relationship graph. *Access rules* are used, which specify the set of *access conditions* under which a certain resource can be accessed. Access conditions are a function of the relationship type, depth and trust level. Relationship certificates based on symmetric-key cryptography are used by a requestor to prove that he satisfies some given access conditions. To access resources held by a node with whom the requestor has no direct relationship, the requestor retrieves from a central node the chain of relationship certificates along the path from the resource owner to himself. Clearly, the central node is a trusted third party, as it knows the relationships of all nodes in the network.

In [9] a mechanism to protect personal information in social networks is described where nodes in the network are anonymous and cannot be linked to a specific users; in contrast, the data and the relationships are public, which might facilitate user re-identification.

An innovative privacy-preserving approach is described in [4] which leans on the access model in [3] and focuses on relationship protection: a user can keep private that he has a relationship of a given type and trust level with another user. Relationship certificates are encrypted and are treated like a resource in their own right: access to a certificate is granted using a *distribution rule* for that certificate, where the *distribution conditions* to be satisfied by users wishing to access the certificate are specified. If a user satisfies the distribution rule for a certificate, he receives the corresponding symmetric *certificate key* allowing him to decrypt the certificate. In [4] a scheme is proposed to manage and distribute certificate keys. Encrypted certificates are stored at a central node; due to encryption, the central node does not have access to the cleartext certificates, so it does not need to be trusted in this respect. However, the central node needs to be trusted in the following aspects: i) trust level computation when several relationship certificates are chained (indirect relationship between a resource requestor and a resource owner); ii) certificate revocation enforcement when a relationship ceases to exist (the central node must maintain a certificate revocation list and inform the other nodes about new revocations).

1.1 Contribution and Plan of This Paper

Enabling private relationships in social networks is an important issue raised in [4]. We describe in this paper a public-key protocol which offers the same features of [4] while eliminating the need for a central node.

Section 2 describes our protocol. A comparative analysis is given in Section 3. Section 4 is a conclusion.

2 A Public-Key Protocol

We follow the framework in [3], that is, we consider that the node owning a resource *rid* (hereafter, the *resource owner*) establishes an access rule $AR = (rid, AC)$ where AC is the set of access conditions to be simultaneously satisfied to access *rid*. Several alternative access rules can be defined for a resource. An access condition is a tuple $ac = (v, rt, d_{max}, t_{min})$ where v is the node with which the node requesting resource *rid* (hereafter, the *requestor*) must have a direct or indirect relationship, and rt, d_{max}, t_{min} are, respectively, the type, the maximum depth and the minimum trust level that the relationship should have. Let PK be the public key of the resource owner and SK be its private key.

We consider that access should be enforced based on the relationship path between requestor and resource owner that yields the maximum trust level. (In [3,4] the trust level is computed taking into account all paths between requestor and resource owner, which is more thorough but might lead to overprotection: a requestor with a highly trusted direct relationship to the owner might be denied access just because there is also a requestor-owner indirect relationship with low trust through a third user.)

We first describe the access control enforcement protocol.

Protocol 1 (Access control enforcement)

1. *The requestor A requests access to a resource rid owned by a resource owner B.*
2. *The resource owner B returns a signed message with all the access rules (say, $AR_1, \ldots, AR_r$) defined for rid, that is $SK(AR_1, \ldots, AR_r)$.*
3. *The requestor A sees to it that the resource owner receives one or several relationship certificates proving that the requestor satisfies all access conditions corresponding to at least one of the access rules. Several cases can be distinguished depending of the depth required:*
   (a) Depth 1. *A and B have a direct relationship, that is, A is related to B through a relationship of type rt and trust level t_{AB} represented by a tuple (A, B, rt, t_{AB}) and B is related to A through a relationship (B, A, rt, t_{BA}). Note that the relevant trust level here is t_{BA} (how much B trusts A) which is assumed to be* unknown *to A. In this case A directly asks B whether he is granted access to the resource on the basis of (B, A, rt, t_{BA}). If B evaluates that rt and t_{BA} satisfy the set of access conditions targeted by A, then A is granted access. Otherwise, A is required to resort to other direct relationships or indirect relationships.*
   (b) Depth 2. *If A and B have no direct relationships (or these are not enough to buy him access) and there are access rules with $d_{max} \geq 2$, then A asks to all users with whom A is directly related whether they have direct relationships of the relevant type rt with B. Assume C is directly related to both A and B with relationship type rt* and is willing to collaborate. *Then C sends to B a signed and encrypted certificate of his relationship $PK_B(SK_C(C, A, rt, t_{CA}))$, where $PK_B(\cdot)$ denotes encryption under the*

public key of B and $SK_C(\cdot)$ denotes encryption under the private key of C. Then C tells A that a certificate was sent to B, but does not reveal *the certificate content. At this point B evaluates whether a relationships of type rt, depth 2 and trust level $t_{CA} \cdot t_{BC}$ is enough to grant access of A to rid.*

(c) Depth 3. *If access at depth less than or equal to 2 cannot be obtained and there are access rules with $d_{max} \geq 3$, then A requests to users C directly related to him to attempt access with depth 2 on A's behalf: each C directly related to A* and willing to collaborate *contacts his other directly related users D about possible direct relationships between D and B (similarly to what A did in Step 3b). If a D with direct relationships to C and B exists and* is willing to collaborate*, B receives a chain of two certificates*

$$(PK_B(SK_C(C, A, rt, t_{CA})), PK_B(SK_D(D, C, rt, t_{DC})))$$

Now B evaluates whether a relationship of type rt, depth 3 and trust level $t_{CA} \cdot t_{DC} \cdot t_{BD}$ is enough to grant access of A to rid.

(d) Successive depths. *In case of failure at depth 3, successive depths are tried in a similar way while access rules are left which accept higher depths.*

Remark 1. There is some privacy price to be paid for using indirect relationships: intermediate users (*e.g.* C, D, etc.) are required to disclose to the resource owner B their trust level in their upstream neighbor (the upstream neighbor of C is A, the upstream neighbor of D is C, etc.). Such a disclosure brings no direct benefit to the intermediate user beyond staying in good terms with the upstream neighbor requesting collaboration. If helping the upstream neighbor is not enough motivation and/or the resource owner B is not trusted enough to be revealed how much the upstream neighbor is trusted, some intermediate nodes might refuse collaboration; this is why we stress the willingness to collaborate as a condition in Protocol 1. However, this shortcoming happens in any social network in which indirect relationships are used for resource access and users are free to decide on their collaborations. □

Remark 2. When the resource owner advertises the access rules for a resource, the access conditions in those rules leak the relationships the owner is involved in (*e.g.* if the owner accepts rt = 'Colleague at company X' this means that he works at Company X). In [4] the relationship type is kept confidential through a symmetric encryption scheme. This becomes tricky when the same relationship type is encrypted using two different keys by two different user communities and these merge at a later stage; another issue is how to revoke the key used to encrypt a given relationship type. An alternative and simpler strategy is to "camouflage" the real relationship types among a large number of bogus relationships; then access conditions are published some of which use real relationship types and most of which use bogus relationship types. A bogus relationship type rt' is one that has never been established by the owner with anyone, so that no

one can request access based on rt'. The advantage is that a snooper cannot tell bogus relationships from real ones, so that he does not know which relationships the owner is actually involved in. □

We will show in the next section that Protocol 1 has the advantage of not requiring any complementary protocols for certificate management or revocation, nor does it require to use any trusted third party or central node.

3 Comparative Analysis

We will compare Protocol 1 with the proposal in [3,4] based on two aspects: protection of relationship certificates and certificate revocation.

3.1 Protection of Relationship Certificates

In the proposal [3,4] relationships are protected through relationship certificates to be presented by a requestor to prove that he has the credentials to access a certain resource. Such certificates are stored at a central node and retrieved from it when needed.

Relationship certificates must be protected because they often convey information that the nodes involved in the relationship would like to keep private (who is related with whom, what the relationship type is and what the trust level is). The authors of [4] propose to view a certificate as a resource and define *distribution rules* and *distribution conditions* to access it; in fact distribution rules and conditions are analogous to the access rules and conditions defined for standard resources. Each certificate is kept encrypted at the central node (who cannot thus read it) under a *certificate key* $CK = (k_{RC}, id_{RC})$ where k_{RC} is a symmetric key and id_{RC} is the corresponding key identifier. When a user satisfies the distribution rule for a certificate, he is given the corresponding certificate key.

The following features of the above model leave room for improvement:

- A central node is needed which might not always be reachable, especially in social networks implemented over ad hoc networks.
- Being symmetrical, certificate keys require a complex key management scheme. In fact, a certificate key distribution algorithm is proposed in [4]: each time a node (user) establishes a new relationship, or when the node receives a new certificate key from one of its neighbors, the algorithm is run by the node to verify whether such a key must be further distributed. Distribution rules related to the certificate are transformed to be relative to the receiving node: the algorithm substitutes the node components of each distribution condition with the identifier of the node receiving the rule, and the depth components are decreased by one unit.

The improvement offered by Protocol 1 can be described as:

Fault tolerance: No central node is required to be reachable all the time. For each resource access, a user tries to get the backing of the nodes with whom

he is related. If one of those nodes is temporarily unreachable, this is not a problem as long as the user can reach other nodes. As mentioned above, access is enforced based on the (reachable) relationship path between requestor and resource owner that yields the maximum trust level; since we do not need all relationship paths between requestor and owner, a certain amount of connectivity failure is tolerable.

TTP-freeness: No central node is required to compute the trust level between requestor and owner. This computation is done by the resource owner on the basis of the received certificates (whereas in [4] the central node is entrusted with such a critical computation).

Minimum relationship disclosure: In Protocol 1 the resource requestor does not see any of the relationship certificates that will be used by the resource owner to decide whether he is granted access. The operation of [4] is different, because there it is the requestor who retrieves the relevant relationship certificates (the certificate chains connecting the requestor to the owner) and forwards them to the owner. This is undesirable: if the retrieved certificates are left unencrypted, then the requestor can see the private information in them; if they are encrypted, then a scheme for managing certificate keys is required (see remarks about certificate key distribution).

3.2 Certificate Revocation

In a social network, users must be able to establish new relationships and revoke some of the existing ones. The use of certificate keys in [4] has implications for revocation. If certificate revocation is not properly managed, unauthorized distribution of certificate keys may occur: if A revokes a certificate relationship with depth 2 relating A to B, it might happen that B continues to distribute the certificate key to his neighbors after revocation. The solution adopted in [4] is to notify revocation to the central node, who removes the revoked certificate from the central certificate directory and stores the certificate identifier in a certificate revocation list (CRL). Thus, the central node actually plays the role of a trusted third party. Besides TTPs being a potential source of conflict, this arrangement requires all users to check the CRL before accepting or distributing a certificate, which causes substantial overhead and implicitly assumes some user trustworthiness.

Protocol 1 has the advantage of being TTP-free. If A wants to access a resource owned by B using an indirect relationship through C and C has decided to revoke his relationship with A, then C will simply send no certificate to B. There is no need to keep revocation lists at a central node.

4 Conclusion and Future Work

Protecting the type and trust level of relationships is an important privacy issue in social networks first raised in the important contribution [4]. We are aware of only one approach in the literature addressing that problem. Being based

on symmetric-key cryptography, that approach requires a central node which must always be reachable and which behaves as a trusted third party. We have presented a public-key protocol that also achieves relationship protection, with the advantages of being fault-tolerant and TTP-free. Besides, the new protocol avoids revealing the content of relationships to the resource requestor and substantially simplifies relationship revocation.

Future work will focus on devising public-key cryptographic techniques to deal with the relationship type leakage associated to access rule advertising by owners. The aim is to gain simplicity with respect to the symmetric-key strategy without having to "litter" access rules with a large number of bogus relationship types.

Disclaimer and Acknowledgments

The author is solely responsible for the views expressed in this paper, which do not necessarily reflect the position of UNESCO nor commit that organization. This work was partly supported by the Spanish Ministry of Education through project SEG2004-04352-C04-01 "PROPRIETAS" and by the Government of Catalonia under grant 2005 SGR 00446.

References

1. Ashri, R., Ramchurn, S.D., Sabater, J., Luck, M., Jennings, N.R.: Trust evaluation through relationship analysis. In: 4th International Joint Conference on Autonomous Agengs and Multiagent Systems (AAMAS 2005), pp. 1005–1011. ACM, New York (2005)
2. Barnes, S.B.: A privacy paradox: social networking in the United States, First Monday, vol. 11(9) (September 2006)
3. Carminati, B., Ferrari, E., Perego, A.: Rule-based access control for social networks. In: Meersman, R., Tari, Z., Herrero, P. (eds.) On the Move to Meaningful Internet Systems 2006: OTM 2006 Workshops. LNCS, vol. 4278, pp. 1734–1744. Springer, Heidelberg (2006)
4. Carminati, B., Ferrari, E., Perego, A.: Private relationships in social networks (IEEE Catalog Number 07EX1627). In: Private Data Management'2007 workshop of the ICDE 2007- 23rd IEEE International Conference on Data Engineering, Istanbul, Turkey, April 15-20, 2007, IEEE Computer Society Press, Los Alamitos (2007)
5. Facebook: `http://www.facebook.com`
6. Sabater-Mir, J.: Towards the next generation of computational trust and reputation models. In: Torra, V., Narukawa, Y., Valls, A., Domingo-Ferrer, J. (eds.) MDAI 2006. LNCS (LNAI), vol. 3885, pp. 19–21. Springer, Heidelberg (2006)
7. Staab, S., Domingos, P., Mika, P., Golbeck, J., Ding, L., Finin, T.W., Joshi, A., Nowak, A., Vallacher, R.R.: Social networks applied. IEEE Intelligent Systems 20(1), 80–93 (2005)
8. Videntity: `http://videntity.org`
9. Wang, D.-W., Liau, C.-J., Hsu, T.S.: Privacy protection in social network data disclosure based on granular computing. In: Proc. of the 2006 IEEE International Conference on Fuzzy Systems (HICSS'05), pp. 997–1003. IEEE Computer Society, Los Alamitos (2006)