An Authorization Infrastructure for Nomadic Computing

Kan Zhang, Tim Kindberg
Mobile Systems and Services Laboratory
HP Laboratories Palo Alto
HPL-2001-228
September 14th , 2001*

E-mail: kan_zhang@hp.com, timothy@hpl.hp.com

We present an infrastructure for flexible and secure access to a group of distributed services in a nomadic computing environment, wherein users access local services from their mobile, wirelessly connected devices. We describe a secure 'hand-off' protocol, which allows a user to register with a single service that “hands off” authorization to access a subset of the services. Our protocol helps maintain the user's privacy. It allows the services (which may be implemented on simple appliances) and the user's mobile device to have modest resources: services do not have to be online to any party except the user's device and the storage and communication requirements are minimal. In addition to the hand-off protocol, the paper presents a model for authorization hand-off and describes related research and technologies.

* Internal Accession Date Only Approved for External Publication
© Copyright Hewlett-Packard Company 2001

An authorization infrastructure for nomadic computing

Kan Zhang &Tim Kindberg

Mobile Systems and Services Laboratory
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304, USA
kan_zhang@hp.com, timothy@hpl.hp.com

Abstract

We present an infrastructure for flexible and secure access to a group of distributed services in a nomadic computing environment, wherein users access local services from their mobile, wirelessly connected devices. We describe a secure 'hand-off' protocol, which allows a user to register with a single service that 'hands off' authorization to access a subset of the services. Our protocol helps maintain the user's privacy. It allows the services (which may be implemented on simple appliances) and the user's mobile device to have modest resources: services do not have to be online to any party except the user's device and the storage and communication requirements are minimal. In addition to the hand-off protocol, the paper presents a model for authorization hand-off and describes related research and technologies.

1 Introduction

We present an infrastructure for flexible and secure access to a group of distributed services in a nomadic computing environment. This work is part of the CoolTown project [1], which is investigating 'nomadic' computing systems: ones in which users, carrying wirelessly connected devices, enter places and use local services associated with those places. The services may be implemented and provided locally by appliances in the place, such as printers, data projectors and soft-drink dispensers. Or they may be local only in the sense that their functionality is related to the location, such as a place-specific information service. Services may be implemented elsewhere but made accessible through physical interfaces in the place such as CoolTown's short-range infrared beacons [1], which emit service addresses for capture by local clients.

We are interested in secure hand-off protocols that facilitate a nomadic user's access to place-specific services. In particular, we are interested in the case that those services are distributed throughout a large area in such a way that they are offline from any central point of administration or control. In nomadic computing, there may be a large number of simple appliances and other services and it is often too costly to implement authorization mechanisms, especially on simple appliances. Some of them simply do not have the necessary resources to run an authorization policy engine by themselves. Even if they do, the cost of keeping the authorization policy consistent on multiple services may be prohibitive. Therefore, in many cases, we need an authorization infrastructure in which there is a centralised authorization policy engine where all the authorization decisions are made.

For example, in Hotel Nomad, all the appliances in a conference room, such as a projector, a printer and a video conferencing facility, together with a service that provides information about local facilities, form a collection under one administration. A conference room manager will make authorization decisions for using those services based on, e.g., whether they have been paid for. The conference room manager or his computerized proxy serves as the centralised authorization policy engine.

Other examples are where a real-estate agent gives a user the right to gain entrance through electronic locks to a selection of homes, or a holiday agent gives the user the right to gain entrance to a selection of chalets and, in each case, use services within those places.

A major difficulty in implementing an authorization infrastructure in a nomadic computing environment is that we cannot assume that the nomadic user or those simple services will have online connections to the centralised authorization policy engine. For example, a hotel guest at Hotel Nomad may not be able to talk to the conference room manager at the time he wants to use the conference room projector. Nor can the manager control the projector remotely. This means we cannot use existing authorization solutions for distributed networks, such as OSF DCE [6] or SESAME [5,7].

In a nomadic computing environment, there is a need for a 'hand-off' protocol such that authorization information can be given to a nomadic user beforehand and verified by the local services off-line. For example, a guest at Hotel Nomad who requires the use of conference room services and has paid for them may be given appropriate authorization credentials, e.g. at the hotel lobby when he checks in. Those credentials can

be stored on his PDA (personal digital assistant), mobile phone or other portable device, and can be used to obtain access to those conference room services that he had paid for. The conference room services should be able to verify those credentials locally.

Any implementation of such an authorization infrastructure must be efficient in terms of both storing credentials and verifying credentials due to the computational and storage limitations of mobile devices and simple appliances. What complicates the problem is that there may be a large number of services and it is generally impossible to predict which services the user will access or in which order. Moreover, different users may be entitled to access different sets of services. This means the credentials have to be short and yet flexible enough for different access patterns.

In addition, it should be possible for the nomadic user to remain anonymous (if desired) when using the authorization service. A nomadic user who has anonymously established a relationship with a place such as a hotel should not be forced to expose their true identity when they obtain or use their credentials for the place's services.

Moreover, it is desirable that a service cannot learn whether its users have the right to access other services. This is based on the principle of 'need-to-know'. For example, a user who accesses services at various shops in a mall may not want individual shops to know which others he or she has an interest in.

Contribution and overview of paper

We give a model that specifies an authorization service for a nomadic computing system that satisfies the above requirements (Section 3). We give a hand-off protocol (Section 5) that implements the model efficiently with respect to time, storage and communication load. Section 2 discusses related research efforts and Section 4 describes related technology that influenced our design including secure multicast group membership and encrypted broadcast. Section 6 concludes.

2 Related work

Some of the earliest work on authorization in distributed systems includes Grapevine [4] where end-servers query registration servers to determine whether a client is a member of a particular group. A similar approach is employed in Sun's Yellow Pages where centrally maintained files such as /etc/group are consulted for authorization purposes. In both cases, although authorization decisions are made locally, the local server has to obtain authorization data online from a remote server.

The Kerberos authorization model for distributed systems is based on the principle that each service knows best who its users should be and what form of authorization is appropriate, so each service manages its own authorization information [2]. The goal of Kerberos authentication is to allow different services to implement different authorization models, and to allow those authorization models to assume that authentication of user identities is reliable. Therefore, each service must determine authorization solely on the basis of the client's identity carried in the service ticket, which means each service must be configured with the identity of all possible users. This leads to management overheads of tracking down and modifying such authorization entries.

The SESAME project [5,7] extends the Kerberos authentication system and defines a scheme for securely propagating principals' privileges, including roles and groups, from clients to application servers in order to reduce the authorization management overhead at application servers. Fairthorne [6] discusses how the SESAME mechanism for secure transmission of privilege can be incorporated into an OSF DCE environment. However, both SESAME and OSF DCE employ online security servers as a direct result of extending Kerberos. Use of online security servers is important in the SESAME architecture because of its emphasis on the ability to facilitate security policies that require constraints and monitoring on who can sign-on to a system even before any access to target applications is allowed.

Several other works proposed off-line authorization solutions, such as delegation certificates in the Digital Distributed System Security Architecture [13,14], the cascaded authentication mechanism by Sollins [12], proxy-based authorization by Neuman [9] and by Trostle and Neuman [11], and authorization certificates by Woo and Lam [15,16].

Those solutions are geared toward authenticated delegation in distributed systems and are not suitable for our nomadic computing model. In distributed computing, intermediate servers often receive high-level requests from initiating clients and perform some series of low-level operations on a number of other target servers. The requests arriving at the target servers appear to be the action of the intermediary rather than the true initiator. In such cases, authenticated delegation allows a client to give verifiable authorization credentials to the intermediate servers so that the target servers can verify that the intermediary is acting on behalf of the client. The basic idea of authenticated delegation used in the above solutions is fairly straightforward. The credentials that a client gives to an intermediary consists of two parts: (1) a tamper-proof certificate which includes the client identifier (or the client's capability), authorization restrictions and an authentication key encrypted for the end-server, and (2) an unencrypted copy of the same authentication key for the intermediary to prove proper possession of the certificate. When requesting service

from the end-server, the intermediary forwards the certificate to the end-server. The end-server then decrypts the authentication key and uses it to verify that the certificate was properly issued to the bearer.

Although the principle of authenticated delegation can be extended to our nomadic authorization problem, existing solutions are not directly applicable. In authenticated delegation, the focus is on passing access rights from the client to the intermediary. When the client identifier is used in a delegation certificate, it is assumed that the identity of the client is already stored on end-servers' access control lists (ACLs). Hence, the delegation certificate only needs to include the client identifier with possible additional restrictions on access rights. Conceptually what the certificate says is that the intermediary is authorized to access any end-servers that the client is entitled to. It is not necessary to specify explicitly which end-servers the intermediary is authorized to access in the certificate. Although the additional restrictions could be different from one end-server to another, how to efficiently encode those restrictions is not seen as an issue and therefore has not been addressed. When a client's capability rather than identity is used in a delegation certificate, the end-server only needs to verify the authenticity of the capability without the need to maintain an ACL. However, in existing systems the capabilities are usually end-server specific. The issues of generating compact capabilities that can be used on a large number of end-servers and in a privacy-preserving way have not received much attention.

In our nomadic authorization model, by contrast, the focus is on giving a nomadic user access rights to a large number of services. Different users may be authorized to access different sets of services. Since these accesses are unpredictable and off-line, a nomadic user should be given an authorization certificate beforehand for all and only those services to which he/she is entitled. These services may be simple and may not have their own ACLs. They trust and expect the central authorization policy engine to inform them to accept or not to accept an access request from a user. We need an efficient way to specify which services the nomadic user is authorized to access. Adding the central authorization policy engine's id to the certificate does not help since that engine is authorized to access all services, not the particular subset that the user is allowed to access.

A naive solution would be to give the nomadic user one certificate for each service to which he/she is entitled. However, such a solution does not scale in our nomadic model, which may contain thousands of simple services and the storage space on the user's mobile device is limited.

The challenge we are facing is how to design an off-line authorization solution that can scale to large number of services and at the same time preserve the user's privacy in the sense defined in Section 1. The authorization certificate has to be short enough so that the nomadic user can store it on his PDA and yet flexible enough to specify in a verifiable way any subset of services should it be the subset that the nomadic user is entitled to.

3 Model

Suppose there are n services s_i, $i=1…n$, in a nomadic computing environment. These services form a service group S, i.e., $i=1…n$, and $|S| = n$. In addition, there is a distinguished service L called "Lobby". Lobby L maintains a central authorization policy engine for all services in S. A user U should first register with Lobby L before he/she can use the services in the service group. Registration does not necessarily involve disclosure of U's true identity. Lobby L will use the authorization policy engine to derive a subset $S_U \subseteq S$ of services that user U can access. Our task is to design an off-line authorization infrastructure that Lobby L can use to enable user U to access any service $s \in S_U$ securely and efficiently later on.

It is an important requirement that S may change, by adding or removing services, at any time and without affecting the existing set-up for other services. There may be thousands of services in our nomadic computing environment and new services are constantly added or old services removed. It would be economically prohibitive if every time we add or remove a service we have to change the set-up of the rest of the services, such as the keys stored on those services. However, we assume that any subset $S_U \subseteq S$ of services that user U can access is fixed once it has been defined.

In our trust model, Lobby L is trusted to behave correctly. Services are also trusted and behave independently. In other words, compromising of one service does not affect the functioning of another service. The operational model of our off-line authorization protocol has the following three phases.

Set-up phase

During the Set-up Phase, we assume Lobby L can communicate securely with each service $s_i \in S$, $i=1…n$, and exchange whatever data is necessary.

User registration phase

During the client registration phase, Lobby L should use the central authorization policy engine to obtain a list of services that user U is entitled to access. For example, the list could be a function of the user's identity or the payment made by an anonymous user. Exactly how Lobby L does this is outside the scope of this work.

Suppose that Lobby L has found out that user U is entitled to access a subset $S_U \subseteq S$ of services. Lobby L

then generates and gives to user U some credential C_U based on S_U. User U stores C_U on his mobile device such as a PDA.

User access phase

When user U later wants to use a service $s \in S_U$, user U uses credential C_U to convince s that U is authorized to access it. A major constraint is that neither U nor s is able to contact Lobby L at the User Access Phase. The verification has to be done off-line. As a by-product, we want U and s to end up with a session key.

In terms of security, we want to achieve the following properties.

- Every $s \in S$ can authenticate whether $s \in S_U$, i.e., s can verify if U is authorized to access s's service.
- User U can authenticate if $s \in S$, i.e., user U can verify if the service he/she is talking to is from service group S.
- No other user U' can access any $s \in S_U$ as U without colluding with U.
- No service $s \in S$ can tell if another service $s' \in S_U$.

One solution is to give each service its own key and compute one credential for each service belonging to S_U. But this requires n credentials to be stored on a user's mobile device where n may be large. Another simple solution is to provide every possible subset of services with a key and give every service the keys corresponding to the subsets it belongs to. This requires every service to store 2^{n-1} keys, which scales even more poorly for large values of n.

In our nomadic computing environment, both computational power and storage capacity at appliances and user's mobile device are rather limited. Of these, the most stringent are the storage space at user's mobile devices and wireless network activity costs. Hence, our solution is geared toward minimizing the size of credential C_U. In doing so, we also minimize the communication complexity between users' mobile devices and services. Storage space at simple appliances could also be very limited. Therefore, we will also pay attention to the storage requirements at services. As to computational power, it is reasonable to assume that user's mobile devices are capable of running symmetric key algorithms and services are at least capable of verifying digital signatures.

4 Related Technology

4.1 Secure multicast group membership

If we view Lobby L as a group controller in a multicast group and services as group members, the user registration phase (not the user access phase) of our off-line hand-off problem becomes very similar to the multiple-leave problem for closed multicast groups. When some members leave a multicast group, the group controller has to inform the rest of the members in the group and give them a new group key (which they use for secure communication). If we think of S as the original multicast group and S_U as the new multicast group after the leave operation, the same protocol used for group key updating can be used to inform the subgroup $S_U \subseteq S$ of services that the user who is contacting them is authorized to access their services. In this view, the user serves as the communication media for multicast, though in an asynchronous way, i.e., the user's accesses are viewed as communicating the group controller's message to group members.

Among various secure multicast protocols, those applicable to our problem are centralized approaches that can exclude a subset k of members without the need to set-up an entirely new group. Chiou and Chen [18] described an interesting method based on the Chinese Remainder Theorem for sending a common key to a selected set of members. The size of the broadcast data is $O(m \log m)$, where $m = n - k$ is the number of members in the broadcast group. Gong [19] describes a key distribution protocol for secure broadcast using polynomial interpolation that achieves similar efficiency in terms of the size of the broadcast data.

Wallner et al. [20] and Caronni et al. [24, 25] proposed hierarchical tree-based schemes for the broadcast exclusion problem where k, the number of receivers excluded, is allowed to be arbitrary. These schemes, however, require $O(k \log n)$ transmission overhead and $O(\log n)$ keys for every receiver. The work of McGrew and Sherman [23], and Canetti et al. [21] shows how to reduce the transmission overhead by a constant factor (a factor of 2 in the case of binary trees).

From our perspective, the above schemes for secure multicast are not very efficient in terms of transmission overhead when k is either very large (the $O(k \log n)$ result) or very small (the $O(m \log m)$ result). The problem of secure multicast group membership and our hand-off problem have different threat models and purposes. In the secure multicast group membership problem, the purpose is transmitting a new group communication key to the remaining subgroup of members while preventing leaving group members from getting the new group key. In our hand-off protocol, the emphasis is on identifying the new group in a secure and privacy-preserving way without distributing a new key. Hence, our hand-off problem can be seen as a simpler problem of identifying the privileged group, i.e., the subgroup $S_U \subseteq S$ that the nomadic user is authorized to access. Ideally, we only need 1 bit of information to represent whether a service is in S_U or not. Hence, our problem should have a more efficient solution than secure multicast in terms of transmission overheads and storage requirement at group members (services).

4.2 Broadcast encryption

Broadcast encryption schemes are also similar in their goals to the user registration phase of our hand-off problem. Broadcast encryption schemes define methods for encrypting content so that only privileged users are able to recover the content from the broadcast. Correspondingly, in the hand-off problem only certain services are 'privileged' by the Lobby to provide their services to the user.

Fiat and Naor [26] were the first to introduce broadcast encryption (in the context of pay-TV). They suggested methods of securely broadcasting key information such that only a selected set of receivers can decrypt this information, while coalitions of up to k other receivers can learn nothing, either in the information-theoretic sense, or under a computational security model.

In broadcast encryption, re-keying is viewed as the most significant cost of the system, and hence is done as infrequently as possible. This is a desirable property for our purpose. However, broadcast encryption schemes are even less efficient than secure multicast schemes in terms of storage at the receiver and transmission length. This remark includes extensions to the basic work of Fiat and Naor [27, 28, 29]. Recently Luby and Staddon [30] studied the trade-off between the transmission length and the number of keys stored in the receivers. A main part of their work is a disappointing lower bound, showing that either the transmission will be very long or a prohibitive number of keys need to be stored in the receivers.

As with secure multicast group membership, one of the central goals of broadcast encryption is the secrecy (and/or integrity) of the messages (or broadcast encryption key). Our secure hand-off protocol is a narrower problem that is only concerned with the secrecy and integrity of the receivers' identities. This important difference allows us to employ the secret set mechanism, which is much simpler than full-blown encryption and which we now describe.

4.3 Secret sets

Molva and Tsudik introduced the notion of a secret set – a basic construct for communication with groups of mutually suspicious entities. A set is secret if any entity can test its membership in the set but can determine neither the other set members nor the cardinality of the set. One of the more efficient constructions of secret sets is using bit vectors. The authors reason that the total amount of information needed to represent a secret set equals the cardinality of that set, i.e., the number of members therein. However, the difficulty in actually reducing the representation of a secret set S_U to $m = |S_U|$ bits is the need to label them somehow. It is easy to see that simply generating a bit vector of length m ($m < n = |S|$) will result in confusion since a potential member has no means to determine the correct bit position (i.e., the bit it should process). Based on the above, the authors conjectured that the minimum length for a bit vector representation of a secret set is the number of all possible members – the cardinality of S.

Secret sets based on symmetric key method

Assuming that a group controller shares a symmetric secret key with every member in the group S ($|S| = n$), the secret set can be computed as follows. The group controller first generates a random number b and set the q-th bit of the n-bit vector V to:

$\text{MSB}(K_q\{b\})$ if group member $q \in S_U$

$1 - \text{MSB}(K_q\{b\})$ otherwise

Where $K_q\{b\}$ denotes the encryption of b with key K_q and MSB(y) denotes the leftmost (most significant) bit of y.

The secret set is (V, b) and its length is ($n + \log b$) bits.

Secret sets based on Diffie-Hellman method

Alternatively, if each group member has a pair of Diffie-Hellman keys and pre-distributes the public exponent to the group controller, the secret set can be generated as follows. Suppose group member q's Diffie-Hellman public exponent is $g^{Kq} \bmod P$ (where g is the base, K_q is the private key and P is a large prime, $P - 1$ being the group order.) The group controller first generates a random number b and set the q-th bit of the n-bit vector V to:

$\text{MSB}(K_q\{g^{bKq}\}(\text{mod } P))$ if group member $q \in S_U$

$1 - \text{MSB}(K_q\{g^{bKq}\}(\text{mod } P))$ otherwise

The resulting secret set is (V, $g^b(\text{mod } P)$) and its length is ($n + \log P$) bits.

5 The hand-off protocol

As we discussed earlier, both secure multicast and broadcast encryption focus on transmitting privileged information to a selected subset of services while preventing the rest from learning it. Our hand-off problem is in fact a much simpler set identification problem. Although a service can infer whether it belongs to the selected subset by testing if it can get the privileged information, applying techniques from secure multicast or broadcast encryption to our hand-off problem is over-kill.

We choose to use secret sets as our set identification method. Secret sets prevent one service from learning whether the user is also authorized to access another service. Moreover, the construction of secret sets is near optimum in terms of its size. If nothing is known about the selected subset S_U, the transmission has to be sufficiently long to uniquely identify the selected

subset S_U. Thus, in general, simply representing a subset $S_U \subseteq S$ requires $|S|$ bits.

We integrate secret sets with techniques from authenticated delegation to obtain our off-line hand-off protocol as follows.

Set-up phase

Suppose there is a group of services s_i, $i=1…n$.

1) Lobby L generates a signature key pair whose public key is denoted as SigKey.
2) Lobby L generates a group key K_G, shared between L and the members of S.
3) Lobby L generates a master key K_M, which it keeps secret, and derives a shared secret key with service s_i as follows [31].

   $$K_i = \text{HASH}(K_M , i, K_M)$$

   where HASH() can be any secure hash function, such as SHA1.
4) Lobby L then sends to each s_i the following in a secure way.

   $$L \rightarrow s_i: K_i, K_G, \text{SigKey}$$

User registration phase

When a nomadic user U comes to register at the Lobby L, L uses the authorization policy engine to obtain the set S_U of services that user U is authorized to access. Lobby L then does the following.

1) Lobby L randomly chooses a number b, and computes a secret set (V, b) using the symmetric key method.
2) Lobby L randomly chooses an authentication key K_U for U.
3) Lobby L computes a signature Sig, verifiable by SigKey, on (V, b, K_U), and builds a credential $C_U = K_G\{V, b, K_U, Sig\}$, i.e. the encryption of $\{V, b, K_U, Sig\}$ using K_G.
4) Lobby L gives to user U the following in a secure way.

   $$L \rightarrow U: K_U, C_U$$

   User U keeps K_U to himself and forwards credential C_U to services when he requires access.

User access phase

1) When user U requires access to service Pi,

   $$U \rightarrow s_i: C_U$$
2) Service s_i decrypts C_U using K_G to get K_U, V, and b and verifies signature Sig using SigKey. If signature verification fails, s_i declines access and aborts.
3) Service s_i checks whether the i-th bit of vector V is equal to MSB($K_i\{b\}$). If not, s_i declines access and aborts.
4) Service s_i and U engage in a mutual authentication protocol using shared authentication key K_U. When successful, s_i provides access to user U. When either party detect an anomaly, they should abort the protocol to protect themselves.

It is easy to see that our hand-off protocol satisfies the security requirements we set out earlier. Only a legitimate service $s \in S$ can decrypt the credential C_U and retrieve its own authorization information from the secret set. Since the secret set is digitally signed by Lobby L, all the legitimate services $s \in S$ can be assured that the retrieved authorization information is authentic. After decrypting the credential C_U, a legitimate service $s \in S$ also obtains an authentication key K_A to be used for mutual authentication with the requesting user. Since only the intended user U can obtain a copy of K_A from Lobby L and user U is never supposed to give out K_A, the mutual authentication makes sure that no other user U' can access the service and that user U is contacting a legitimate service $s \in S$. Finally, the construction of secret sets prevents one service from getting authorization information regarding another service.

Since each service checks a unique bit in the bit vector V to find out authorization information independently, adding or removing a service will not affect other services. The size of vector V simply grows linearly with the number of services. Those bits in bit vector V corresponding to removed services can be reused by new services. The storage requirements at both Lobby L and services are very small, i.e., two symmetric keys and one public key. The size of credential C_U is n bits plus a small constant (less than 100 bytes if using a 512-bit RSA signature).

If the services are capable of performing Diffie-Hellman key exchange, we can assign each service a Diffie-Hellman key and generate the vector V using the Diffie-Hellman method. The size of credential C_U is similar to that using the symmetric key method. Using the Diffie-Hellman method provides the added advantage that any service can potentially become a Lobby if it has access to the authorization policy engine, without the need to store pair-wise keys at the services. This kind of set-up is useful when a user puts a different degree of trust in different services. For example, a number of shops in a shopping mall may form a service group and each of them may function as a Lobby. A user who trusts Tom's Coffee more than any other store, may pay Tom's Coffee to get access to other stores while avoiding dealing with other shops directly.

6 Conclusion

We have described a secure hand-off authorization protocol designed for nomadic computing systems. In such systems, users access a selection of potentially many services. Users are able to obtain authorization by first receiving a certificate from a 'Lobby' service. We have chosen a name that is suggestive of a good location for the service in a place such as a building or a hotel. But it may be placed at any secure point where users can be expected to pass conveniently.

The services in a nomadic computing environment may be provided by relatively simple appliances. In recognition of this, our protocol allows services to be offline from any point of administration at the time of user authorization. Our protocol also has a storage requirement at the services that is constant and modest (two keys). It has a storage requirement at the user's device and a message size requirement that is also modest: *n* bits (where *n* is the total number of services) plus a small amount.

The protocol enables users to maintain their privacy by limiting the information held at services on a 'need-to-know' basis. Services do not store user identifiers and no service can tell whether a given user has access to any other member of the service group.

Our protocol gives authorization for a set of services that is fixed at the time of user registration, whereas in some cases it may be desirable to give users access to a set of services that is only logically identified when the user registers and whose membership may change at run-time. In particular, we do not provide a mechanism for revoking access to particular services in this model. However, it is a simple matter to insert an expiry time or a limit on the number of uses securely into the certificates issued to users, which can be checked by individual services in the user access phase.

Acknowledgements. This paper benefited from discussions with Gita Gopal, Narendar Shankar, John Barton and Jean Tourrilhes.

References

[1] Tim Kindberg, John Barton, Jeff Morgan, Gene Becker, Ilja Bedner, Debbie Caswell, Phillipe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, Celine Pering, John Schettino, Bill Serra. People, Places, Things: Web Presence for the Real World, in Proceedings of WMCSA2000.

[2] S.P. Miller, B. C. Neuman, J. I. Schiller, and J.H. Saltzer. Section E.2.1: *Kerberos Authentication and Authorization System*. Project Athena Technical Plan, MIT Project Athena, Cambridge, Massachusetts, October 1988. (Version 4)

[3] John Kohl and B. Clifford Neuman. The Kerberos Network Authentication Service (Version 5). Internet Request for Comments RFC-1510. September 1993.

[4] Andrew D. Birrell, Roy Levin, Roger M. Needham, and Michael D. Schroeder. Grapevine: An exercise in distributed computing. Communications of the ACM, 25(4):260-274, April 1982.

[5] P. V. McMahon, SESAME V2 Public Key and Authorization Extensions to Kerberos. Proceedings of the 1995 Symposium on Network and Distributed System Security, pp-114-131, February 1995.

[6] S. B. Fairthorne, "Security Extensions for DCE 1.1", OSF DCE RFC 19.

[7] T. Parker, and D. Pinkas, "SESAME V4 -- Overview", Dec 1995. Available as http://www.esat.kuleuven.ac.be/cosic/sesame/doc-txt/overview.txt

[8] Marlena E. Erdos and Joseph N. Pato. Extending the OSF DCE Authorization System to Support Practical Delegation. In *Proceedings of the 1993 PSRG Workshop on Network and Distributed System Security*, February 1993.

[9] B. Clifford Neuman. Proxy-Based Authorization and Accounting for Distributed Systems. In *Proceedings of the 13th International Conference on Distributed Computing Systems*, pages 283-291, May 1993.

[10] Joseph N. Pato, *DCE Authorization Services -- Privilege Server*, OSF DCE Specifications, 1990.

[11] Jonathan T. Trostle and B. Clifford Neuman. A Flexible Distributed Authorization Protocol, In Proceedings of the 1996 Symposium on Network and Distributed System Security, 1996

[12] Karen R. Sollins, *Cascaded Authentication*, in *Proceedings of the 1988 IEEE Symposium on Security and Privacy*, IEEE Computer Society, 1988.

[13] M. Gasser, A. Goldstein, C. Kaufman, and B. Lampson. The Digital distributed system security architecture. In Proceedings of the 1989 National Computer Security Conference, pages 305-319, October 1989.

[14] M. Gasser and E. McDermott. An architecture for practical delegation in a distributed system. In Proceedings of the 1990 IEEE Symposium on Security and Privacy, pages 20-30, May 1990.

[15] T.Y.C. Woo and S.S. Lam. A framework for distributed authorization. In Proceedings of 1st ACM Conference on Computer and

Communications Security, pages 112-118, Fairfax, Virginia, November 3-5, 1993.

[16] T.Y.C. Woo and S.S. Lam. Designing a Distributed Authorization Service, In Proceedings of IEEE INFOCOM'98, March 1998.

[17] O. Rodeh, K. P. Birman, and D. Dolev. *Optimized group rekey for group communication systems*. In Symposium Network and Distributed System Security, Febuary 2000.

[18] G.H. Chiou and W.T. Chen. Secure Broadcast Using the Secure Lock, IEEE Transactions on Software Engineering, 15(8):929-934, August 1989.

[19] L. Gong. New Protocols for Third-Party-Based Authentication and Secure Broadcast, In Proceedings of the 2nd ACM Conference on Computer and Communications Security, pp176-183, Fairfax, Virginia, November 1994.

[20] D. Wallner, E. Harder, and R. Agee. Key Management for Multicast: Issues and Architectures. RFC 2627, June 1999.

[21] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. Proc. IEEE INFOCOM'99.

[22] R. Canetti, T. Malkin, and K. Nissim. Efficient communication-storage tradeoffs for multicast encryption. Proc. EUROCRYPT'99.

[23] D.A. McGrew and A.T. Sherman. Key Establishment in Large Dynamic Groups Using One-Way Function Trees. Technical Report No. 0755, TIS Labs at Network Associates, Inc., Glenwood, MD, May 1998.

[24] G. Caronni, M. Waldvogal, D. Sun, and B. Plattner. Efficient Security for Large and Dynamic Multicast Groups. In Workshop on Enabling Technologies, WETICE 98. IEEE Computer Society Press, 1998.

[25] M. Waldvogal, G. Caronni, D. Sun, N. Weiler, and B. Plattner. The VersaKey Framework: Versatile Group Key Manangement. IEEE Journal on Selected Areas in Communication, 17(8):1614-1631, August 1999.

[26] A. Fiat and M. Naor. Broadcast Encryption. In Proc. CRYPTO'93, LNCS 773, pp 480-491, 1994.

[27] C. Blundo and A. Cresti. Space requirements for broadcast encryption. In Proc. EUROCRYPT'94, LNCS 950, pp 287-298, 1994.

[28] C. Blundo, L.A. Frota Mattos, and D.R. Stinson. Generalized Beimel-Chor schemes for broadcast encryption and interactive key distribution. Theoretical Computer Science, 200(1-2):313-334, 1998.

[29] D.R. Stinson and T. van Trung. Some new results on key distribution patterns and broadcast encryption. Designs, Codes and Cryptography, 14(3):261-279, 1998.

[30] M. Luby and J. Staddon. Combinatorial bounds for broadcast encryption. In Proc. EUROCRYPT'98, LNCS 1403, pp 512-526, 1998.

[31] U. Blumenthal, N. C. Hien, and B. Wijnen. Key Derivation for Network Management Applications. IEEE Network Magezine, pp 26-29, May/June 1997.