A Security Architecture for Tenet Scheme 2

Rolf Oppliger
Amit Gupta
Mark Moran
Riccardo Bettati

Abstract. The bandwith requirements of interactive multimedia applications are exhaustive, causing network congestion to be a major problem. One way to deal with this problem is to use a resource reservation scheme, such as e.g. Tenet Scheme 2. This paper proposes a security architecture for Tenet Scheme 2. The basic ideas are to use Internet layer security protocols, such as the IP Security Protocol (IPSP) and Internet Key Management Protocol (IKMP), to establish authentic communication channels between RCAP daemons, to handle client authentication and authorization locally, and to use a proxy-based mechanism to distribute access rights for target sets and channels. The security architecture uses as its building blocks a collision-resistant one-way hash function and a digital signature system.

1 Introduction

To cover the future needs of business and residential customers in the information age, interactive multimedia applications are envisaged or being developed and installed. But the bandwith requirements of interactive multimedia applications are exhaustive, causing network congestion to be a major problem. One way to deal with this problem is to use a resource reservation and billing scheme that allows particular users to obtain preferential access to network resources, under the control of an admission control mechanism. Permission to make a reservation will depend both upon the availability of the requested resources along the path of the data, and upon satisfaction of policy rules. Several resource reservation schemes and corresponding protocols have been proposed so far. Examples are the resource reservation setup protocol (RSVP) [1] and the Internet stream protocol version 2 (ST-II), as well as the Tenet realtime protocols.

However, the use of a resource reservation scheme requires security to be considered with care. A cryptographic message origin authentication mechanism has been added to the specification of RSVP [2], and this paper proposes a security architecture for Tenet Scheme 2. The basic ideas are

- to use Internet layer security protocols, such as the IP Security Protocol (IPSP) and Internet Key Management Protocol (IKMP), to establish authentic communication channels between RCAP daemons,
- to handle client authentication and authorization locally, and
- to use a proxy-based mechanism to distribute access rights for target sets and channels.

The security architecture uses as its building blocks a collision-resistant one-way hash function and a digital signature system. The rest of this paper proceeds as follows: The Tenet Scheme is briefly introduced in section two. The terminology that is used to subsequently describe the security architecture for Tenet Scheme 2 is introduced in section three. The evolving protocol standards that can be used to provide Internet layer security are overviewed in section four, and the security architecture for Tenet Scheme 2 is described and fully discussed in section five. Conclusions are finally drawn in section six.

2 Tenet Scheme

The Tenet Group at the University of California and the International Computer Science Institute (ICSI) in Berkeley has been working since 1988 to provide practical solutions to the problem of realtime communications [3]. The initial goal of the work was to devise and specify a set of algorithms that, when implemented in a network, would enable the network to offer a realtime communication service to its clients. Such a set of algorithms was called a realtime communication scheme, or Scheme for brevity. Meanwhile, Tenet Scheme 1 has been embodied in a suite of realtime protocols, namely the Tenet Suite 1[1].

In the Tenet realtime protocol suite, the task of channel setup is performed by the Real Time Channel Administration Protocol (RCAP). RCAP is a control protocol that takes client requests containing traffic descriptions and performance requirements and sets up realtime channels accordingly. The data transfers are done by the Realtime Internet Protocol (RTIP) which schedules datagrams according to the resource reservations made by RCAP. At the transport layer, the Tenet Suite consists of two protocols, namely

- the Realtime Message Transport Protocol (RMTP), which is intended for message-based realtime transport between endpoints, and
- the Continuous Media Transport Protocol (CMTP), which offers a stream-based interface and a time-driven mechanisms for applications, such as audio and video transmission.

The protocol that controls data transfers, primarily reacting to the detection of error conditions, is called the Realtime Control Message Protocol (RTCMP). RTCMP performs functions similar to those of ICMP in the Internet protocol suite.

The Tenet Group is currently working on Scheme 2. The main focus of this work is to extend the basic realtime communication service provided by Tenet Suite 1 in two respects [4, 5, 6]:

- to provide abstractions and techniques for efficient multi-party realtime communication;

[1] Further information on Tenet Suite 1 can be obtained in the World Wide Web by following the URL `http://tenet.berkeley.edu/tenet-software.html`.

– to make the client-service interface more flexible.

In Tenet Scheme 2, the key networking abstraction is the target set, which is similar to the host group in IP multicast [7, 8]. Receivers join a target set depending on their interest in a certain transmission, and channels are established from data sources to the members of a target set accordingly. Join and leave primitives support dynamic membership in target sets, as well as the associated change in multicast channels.

From the security point of view, the fact that RCAP uses TCP/IP connections to exchange messages between clients and RCAP daemons must be considered with care. The use of the TCP/IP protocols does neither allow RCAP daemons to authenticate client requests, nor to protect the authenticity and integrity of messages in exchange. As a matter of fact, the TCP/IP protocol suite have been known to be vulnerable and exposed to various attacks, and it is only due to the emerging use of the Internet for commercial purposes that TCP/IP and Internet security have recently become an issue [9].

3 Terminology

A principal refers to a human user or system entity that is registered in and authenticatable to a system. Authentication refers to the process of verifying the claimed identity of a principal, authorization to the process of determining the (access) rights of this principal, and access control to the process of enforcing these access rights. With regard to Tenet Scheme 2, users, clients, and RCAP daemons are considered as principals, and access must be controlled with regard to target sets and channels.

The following notation is used in this paper to refer to users, clients, RCAP daemons, target sets, and channels:

- U_i is used to refer to user $i (i \geq 1)$.
- Every user U_i may have several clients running on his behalf, and C_{ij} is used to refer to client j of user $U_i (i, j \geq 1)$.
- D_i is used to refer to RCAP daemon $i (i \geq 1)$.
- TS_i is used to refer to target set $i (i \geq 1)$.
- Every target set TS_i may have several channels associated with it, and Ch_{ij} is used to refer to channel j of target set $TS_i (i, j \geq 1)$.

A protocol specifies the format and relative timing of messages exchanged between communicating parties. A cryptographic protocol is a protocol that uses cryptographic techniques, meaning that all or parts of the messages are encrypted on the sender' side, and decrypted on the receiver' side. Both the Internet layer security protocols and the protocols that are used for client authentication, authorization, and access rights distribution represent cryptographic protocols. The reader of this paper is thus assumed to be familiar with both cryptography, and the use of cryptographic protocols in computer networks and distributed systems [10, 11]. The following notation is used to describe cryptographic protocols:

- K is used to denote a secret key, and the term $\{m\}K$ is used to refer to a message m that is encrypted with K. The same key is used for decryption, so $\{\{m\}K\}K$ equals m. If K is used to compute a message authentication code (MAC), then the term $\langle m\rangle K$ is used to refer to a MAC computed for message m and key K. An efficient way to compute and verify a MAC is to use a collision-resistant one-way hash function, such as MD4 [12], MD5 [13] or the Secure Hash Standard (SHS) [14], and to key it with a secret key as described e.g. in [15, 16].
- (k, k^{-1}) is used to denote a public key pair, with k referring to the public key, and k^{-1} referring to the corresponding private key. The term $\{m\}k$ is used to refer to a message m that is encrypted with a public key k. The message can be decrypted only with the corresponding private key k^{-1}, so $\{\{m\}k\}k^{-1}$ equals m. In a digital signature system, such as RSA [17], ElGamal [18], or the Digital Signature Standard (DSS) [19], the user's private key is used to digitally sign messages, and the corresponding public key is used to verify the signatures. In this case, the term $\{m\}k^{-1}$ is used to refer to a digital signature giving message recovery, and the term $\langle m\rangle k^{-1}$ is used to refer to a digital signature with appendix [20]. In the latter case, $\langle m\rangle k^{-1}$ abbreviates $(m, \{h(m)\}k^{-1})$, with h being a collision-resistant one-way hash function. The signature can be verified only with the public key k.

In either case, key subscripts may be used to indicate principals, target sets, or channels.

4 Internet Layer Security Protocols

With regard to the Internet' security concerns, both the Internet Research Task Force (IRTF) and the the Internet Engineering Task Force (IETF) have launched corresponding activities. The IRTF Privacy and Security Research Group (PSRG) is adapting the OSI security architecture for the Internet, and the IETF has chartered an Internet Protocol Security Protocol (IPSEC) Working Group (WG) to standardize an Internet layer security protocol. As a matter of fact, the IETF IPSEC WG seeks to standardize both an IP Security Protocol (IPSP) and an Internet Key Management Protocol (IKMP).

4.1 IPSP

The idea of having a standardized network layer security protocol is not new, and several protocols had been proposed even before the IPSEC WG started to meet:

- The Security Protocol 3 (SP3) was proposed by the National Security Agency (NSA) and the National Institute of Science and Technology (NIST) as part of the Secure Data Network System (SDNS).

- The Network Layer Security Protocol (NLSP) was proposed by the International Organization for Standardization (ISO) to secure the Connectionless Network Protocol (CLNP) [21].
- The Integrated NLSP (I-NLSP) was proposed to provide security services for both IP and CLNP.
- SwIPe was yet another network layer security protocol proposal [22].

In spite of their different names and specifications, all of these protocols have one thing in common: they all use IP encapsulation as their enabling technique. IP encapsulation allows IP datagrams to be encrypted and enclosed in outer IP headers. Based on these outer IP headers, the datagrams are then routed through an internet. At a peer systems, the outer IP headers are stripped off, and the IP datagrams are decrypted and forwarded to their final destinations.

The current version of IPSP uses IP encapsulation as its enabling technique, too. In particular, the IPSP suggests the use of two security mechanisms that may be used together or separately:

- The Authentication Header (AH) provides data origin authentication services for IPSP datagrams. In essence, the sender of a datagram computes a MAC over the constant parts of the datagram, and sends the result as AH together with the datagram to the receiver. The receiver extracts the AH, recalculates the MAC, and verifies, whether the value matches the AH that he has originally received from the sender.
- The Encapsulating Security Payload (ESP) uses IP encapsulation to provide confidentiality services for IPSP datagrams.

The IETF IPSEC WG has proposed keyed MD5 [13] as a default algorithm for the AH mechanism, and the Data Encryption Standard (DES) in Cipher Block Chaining (CBC) mode [23, 24] for the ESP mechanism. However, export, import and use of encryption may be regulated in some countries, and other algorithms and modes may be used instead.

4.2 IKMP

With regard to a possible IKMP standard, several proposals have been submitted to the IETF IPSEC WG for further consideration:

- The Modular Key Management Protocol (MKMP) uses long-term master keys to derive short-term session keys that provide perfect forward secrecy [25].
- The Simple Key-Management for Internet Protocols (SKIP) uses implicitly shared long-term Diffie-Hellman keys to derive keys on a per-session or per-datagram basis [26].
- The Photuris[2] Key Management Protocol combines a Diffie-Hellman key exchange with a subsequent exchange of RSA signatures.

[2] "Photuris" is the latin name for the firefly, and "Firefly" is in turn the name for a classified key exchange protocol designed by the NSA for the STU-III secure telephone.

Quite recently, the Internet Drafts related to the IPSP have been approved by the Internet Engineering Steering Group (IESG) as Proposed Standards for the Internet in RFC 1825 to 1829. With regard to the IKMP, the focus of the IETF IPSEC WG is on the Photuris proposal. As a matter of fact, the MKMP proposal has been gathered up in favor of an enhanced version of the Photuris proposal named Photuris Plus or SKEME (Secure Key Exchange Mechanism) [27].

5 Security Architecture

It has already been mentioned previously that the basic ideas of the security architecture for Tenet Scheme 2 are to use Internet layer security protocols to establish authentic communication channels between RCAP daemons, to handle client authentication and authorization locally, and to use a proxy-based mechanism to distribute access rights for target sets and channels. Having overviewed the Internet layer security protocols that can be used to establish authentic channels between RCAP daemons in section four, this section addresses the remaining questions, namely how to authenticate and authorize clients locally, and how to distribute access rights for target sets and channels to other clients. The assumptions are as follows:

- Every RCAP daemon D_i holds a long-term public key pair $(k_{D_i}, k_{D_i}^{-1})$ that can be used in a digital signature system. Note that the public key pair may (but need not) be the same as the public key pair that is used for any Internet layer key management protocol. Also note that key revocation is considered as a problem that is not necessarily related to Tenet Scheme 2, and thus not discussed in this paper.
- Every user U_i (and thus every client C_{ij} that acts on U_i's behalf) is associated with a particular RCAP daemon D_k, and this RCAP daemon is referred to as U_i's (and C_{ij}'s) home RCAP daemon.
- Every RCAP daemon D_i has a list of users who are authorized to have D_i create target sets. The acronym UAL (User Authorization List) is used to refer to this list.

With regard to the last two assumptions it should be made explicit that the notion of a home RCAP daemon has been introduced to avoid the necessity of having and maintaining a global naming scheme. In this case, a RCAP daemon must only know its local users. However, the security architecture as described in this section does not suffer any fundamental change, if home RCAP daemons are clustered or made obsolete by introducing a global naming scheme.

5.1 Authentication

If user U_i has client C_{ij} contact his home RCAP daemon D_k, C_{ij} and D_k must authenticate each other. In principle, C_{ij} has to show that he's acting on U_i's

behalf, and D_k has to show that he's U_i's (and thus C_{ij}'s) home RCAP daemon. There are several authentication protocols available today that can be used for this task. An overview is given in [28]. The protocols require either a secret to be shared between U_i and D_k, or the use of public key cryptography:

- With regard to the first possibility, the secret shared between U_i and D_k may be a personal identification number (PIN) or a password. In this case, a single sign-on mechanism can be used to authenticate a user based on a secret that is comparably weak, and provide him with a cryptographically strong session key.
- With regard to the second possibility, both U_i and D_k must have a public key pair, of which they keep the private key secret and make the public key available to the others. Note that D_k already fulfills these requirements, as $k_{D_k}^{-1}$ is D_k's private key and k_{D_k} is assumed to be publicly available. Consequently, the problem is only relevant for U_i. One possibility to solve the problem is to equip U_i with a personal token that stores $k_{U_i}^{-1}$ on his behalf. However, this solution requires the use of special hardware devices, such as chipcards or PCMCIA cards, and the cost of this solution may be prohibitive. A more realistic approach is either to locally store $k_{U_i}^{-1}$ encrypted with a password-derived key encryption key, or to have D_k store $k_{U_i}^{-1}$, and to use a password-based mechanism to control access to this key.

Note that some operating systems offer a possibility that, in principle, would allow RCAP daemons to authenticate users without having to store additional authentication information. For example, if D_k had read access to the password file of a UNIX system, the entries of this file could be used to authenticate users and clients. Taking advantage of this possibility would simplify authentication considerably. However, it is only fair to mention that it would also expose the users' passwords to malicious software attacks from the RCAP daemon. One way to deal with this problem is to have the RCAP daemon authenticate to a client, before the client requests and delivers the user's password.

Having discussed the various approaches for authentication, it is assumed that a corresponding protocol is used, and that this protocol provides C_{ij} and D_k with a strong session key K_{ik}. This key can then be used for the lifetime of a session to authenticate the origin of messages, and to establish authentic communication channels between clients and RCAP daemons accordingly.

5.2 Authorization and Access Control

To create a target set TS_l on U_i's behalf, C_{ij} has to randomly select a public key pair (k_l, k_l^{-1}), and request D_k to create TS_l by using the following protocol:

$$1 : C_{ij} \longrightarrow D_k \ : \text{Target_Set_Create_Request}(\langle TS_l, k_l \rangle K_{ik})$$
$$2 : D_k \longrightarrow C_{ij} \ : \text{Target_Set_Create_Confirmation}(\langle TS_l, k_l \rangle k_{D_k}^{-1})$$

In step 1, C_{ij} sends a target set create request message to D_k. The message includes TS_l and k_l, as well as a MAC to protect message origin authenticity. C_{ij}

uses K_{ik} to compute the MAC, and D_k uses the same key to verify the MAC. If the MAC is valid, D_k assumes the request to be authentic. If U_i is enumerated in D_k's UAL, D_k creates the target set, and returns a target set create confirmation message to C_{ij} in step 2. Note that the message in this case represents a digital signature with appendix for TS_l and k_l, generated with D_k's private key $k_{D_k}^{-1}$. D_j and C_{ik} both store the message for backup purposes. Finally, C_{ik} announces C_{ij}, D_k, and $\langle TS_l, k_l \rangle k_{D_k}^{-1}$ in public.

If a client C_{xy} is to perform a specific operation on TS_l, such as join or leave, or create a channel on TS_l, he must be authorized accordingly. In principle, he must be granted a proxy that provides him with the corresponding (access) rights by C_{ij}. Therefore, C_{xy} randomly selects a public key pair (k, k^{-1}), and uses the following protocol to request a proxy:

$$\begin{aligned} 1 &: C_{xy} \longrightarrow C_{ij} \;\; : \text{Target_Set_Proxy_Request}(TS_l, \{R, C_{xy}, k\}k_l) \\ 2 &: C_{ij} \longrightarrow C_{xy} : \text{Target_Set_Proxy_Confirmation}(TS_l, \langle R', C_{xy}, k \rangle k_l^{-1}) \end{aligned}$$

In step 1, C_{xy} sends a target set proxy request message to C_{ij}. The message includes TS_l, an encoded set R of requested access rights, C_{xy}, and the public key k. R, C_{xy}, and k are encrypted with k_l which is the public key of the target set TS_l. After having received the message, C_{ij} decrypts the encrypted part with k_l^{-1}, and decides whether he wants to grant access rights to C_{xy}. If he does, he selects a corresponding set R' of access rights, and returns a target set proxy confirmation message to C_{xy} in step 2. The message includes TS_l, R', C_{xy}, and k, with the last three components being digitally signed with k_l^{-1}.

Note that, in general, C_{xy} and C_{ij} needn't be associated with the same RCAP daemon. If U_x and C_{xy} are associated with D_z, and U_i and C_{ij} with D_k, then the target set proxy request and confirmation messages must pass along a path between D_z and D_k, and this path may include several RCAP daemons. The messages that are exchanged between RCAP daemons are assumed to be protected by using the IPSP AH mechanism. Provided that an IPSP connection can be established between D_z and D_k the protocol to request a target set proxy is as follows:

$$\begin{aligned} 1 &: C_{xy} \longrightarrow D_z \;\; : \text{Target_Set_Proxy_Request}(\langle TS_l, \{R, C_{xy}, k\}k_l \rangle K_{xz}) \\ 2 &: D_z \longrightarrow D_k \;\; : \text{Target_Set_Proxy_Request}([[TS_l, \{R, C_{xy}, k\}k_l]]) \\ 3 &: D_k \longrightarrow C_{ij} \;\; : \text{Target_Set_Proxy_Request}(\langle TS_l, \{R, C_{xy}, k\}k_l \rangle K_{ik}) \\ 4 &: C_{ij} \longrightarrow D_k \;\; : \text{Target_Set_Proxy_Confirmation}(\langle TS_l, \langle R', C_{xy}, k \rangle k_l^{-1} \rangle K_{ik}) \\ 5 &: D_k \longrightarrow D_z \;\; : \text{Target_Set_Proxy_Confirmation}([[TS_l, \langle R', C_{xy}, k \rangle k_l^{-1}]]) \\ 6 &: D_z \longrightarrow C_{xy} : \text{Target_Set_Proxy_Confirmation}(\langle TS_l, \langle R', C_{xy}, k \rangle k_l^{-1} \rangle K_{xz}) \end{aligned}$$

The target set proxy request message is passed from C_{xy} to D_z in step 1, from D_z to D_k in step 2, and from D_k to C_{ij} in step 3. In return, the target set proxy confirmation message is passed from C_{ij} to D_k in step 4, from D_k to D_z in step 5, and from D_z back to C_{xy} in step 6. The authenticity of the messages that are exchanged between clients and RCAP daemons are protected by MACs, and the authenticity of the messages that are exchanged between RCAP daemons

are protected by the IPSP AH mechanism. The latter protection mechanism is indicated by using double square brackets in steps 2 and 5.

If C_{xy} wants to use the proxy that he has been granted by C_{ij}, he contacts D_k with the following challenge response protocol:

$$
\begin{aligned}
&1 : C_{xy} \longrightarrow D_k \;\; : \text{Target_Set_Access_Request}(TS_l, C_{xy}, \langle R', C_{xy}, k\rangle k_l^{-1}) \\
&2 : D_k \;\; \longrightarrow C_{xy} : \text{Target_Set_Access_Request_Challenge}(N) \\
&3 : C_{xy} \longrightarrow D_k \;\; : \text{Target_Set_Access_Request_Response}(\{N\}k^{-1})
\end{aligned}
$$

Note that the protocol is given in its short form, without going through all intermediate RCAP daemons between C_{xy} and D_k. In step 1, C_{xy} provides D_k with TS_l, C_{xy}, and the digitally signed part of the target set proxy that he has received from C_{ij} for TS_l. D_k can compare the client information in the request with the client information in C_{ij}'s confirmation. If they match, D_k assumes the access request to be valid. He challenges C_{xy} with a fresh nonce N in step 2, and C_{xy} is to answer with a digital signature for N in step 3. Note that C_{xy} can generate the correct response only if he holds the private key k^{-1} that corresponds to k. If C_{xy} is able to correctly respond with $\{N\}k^{-1}$, D_k assumes that the proxy has indeed been issued for C_{xy}.

In Tenet Scheme 2, a target set may have several channels associated with it, and the right to create a channel on a target set must be granted by the owner of the target set. Consequently, if C_{ij} is the owner of target set TS_l, C_{xy} has to be granted a proxy to create a channel Ch_{lk} on TS_l by C_{ij}. If C_{ij} provides C_{xy} with such a proxy, C_{xy} can use it to have D_k set up Ch_{lk}. In this case, D_k registers C_{xy} with the public key k' as owner of the channel. If another client C_{vw} wants to access channel Ch_{lk}, he has be provided with a proxy by C_{xy}. In this case, C_{xy} uses k'^{-1} to issue and digitally sign proxies for Ch_{lk}. The protocol to request a proxy for a channel is similar to the protocol to request a proxy for a target set. C_{vw} randomly selects a public key pair (k'', k''^{-1}), and uses the following protocol:

$$
\begin{aligned}
&1 : C_{vw} \longrightarrow C_{xy} \;\; : \text{Channel_Proxy_Request}(Ch_{lk}, \{R, C_{vw}, k''\}k') \\
&2 : C_{xy} \longrightarrow C_{vw} \;\; : \text{Channel_Proxy_Confirmation}(Ch_{lk}, \langle R', C_{vw}, k''\rangle k'^{-1})
\end{aligned}
$$

Again, the connection between C_{vw} and C_{xy} may lead through intermediate RCAP daemons, and to keep the description simple, the protocol is given in its short form. In step 1, C_{vw} sends a channel proxy request message to C_{xy}. The message includes Ch_{lk} and $\{R, C_{vw}, k''\}k'$. Note that only the legitimate owner of the Ch_{lk} holds the private key k'^{-1}, and can decrypt the second part of the message accordingly. In step 2, C_{xy} returns a channel proxy confirmation message to C_{vw}. The message includes Ch_{lk}, as well as R', C_{vw}, and k'', digitally signed with k'^{-1}. C_{vw} can use the channel proxy by using the following challenge response protocol:

$$
\begin{aligned}
&1 : C_{vw} \longrightarrow D_k \;\; : \text{Channel_Access_Request}(Ch_{lk}, C_{vw}, \langle R', C_{vw}, k\rangle k'^{-1}) \\
&2 : D_k \;\; \longrightarrow C_{vw} : \text{Channel_Access_Request_Challenge}(N) \\
&3 : C_{vw} \longrightarrow D_k \;\; : \text{Channel_Access_Request_Response}(\{N\}k''^{-1})
\end{aligned}
$$

This protocol is analogous to the protocol that is used to use target set proxies. In step 1, C_{vw} sends a channel access request message to D_k. The message includes Ch_{lk} and C_{vw}, as well as the channel proxy $\langle R', C_{vw}, k\rangle k'^{-1}$. D_k knows the public key k', and can verify the digital signature that is appended to the message accordingly. In step 2, D_k challenges C_{vw} with a nonce N, and in step 3, C_{vw} responds with $\{N\}k''^{-1}$, which is a digital signature for that nonce.

The proxy-based authorization mechanism proposed in this paper is similar to the restricted proxies described in [29]. Both mechanisms use a certificate that carries a public key, and a corresponding private key that links the proxy to its legitimate owner. The main difference between the two mechanisms is related to the way in which the private key is selected and distributed. Whereas in the mechanism described in this paper, the client that requests a proxy selects a public key pair and has the public key digitally signed by the proxy grantor, it is the grantor that selects a public key pair and provides the client with both the certificate and the corresponding private key in [29]. It is assumed that the mechanism described in this paper is advantageous in situations that can't assume secret channels to be available between clients, as well as in situations in which non-repudiation services must be provided, too. Note that if the proxy grantor selects the client's private key, there is, in general, no possibility to provide non-repudiation services. On the other hand, it is assumed that in situations that require a distribution of many proxies to various clients, the mechanism proposed in [29] has scalability advantages. It is, in general, more convenient for a client to be provided with the proxies he needs, instead of having to interactively request them. Nevertheless, it should be noted that both mechanisms need further investigation. In fact, the mechanisms are not mutually exclusive, and both mechanisms could be used in Tenet Scheme 2. For example, one possibility would be to use the mechanism proposed in [29] to distribute bearer proxies that can be used by anyone, and to use the mechanism described in this paper to distribute delegate proxies that can be used only by the clients they name.

6 Conclusions

The bandwith requirements of interactive multimedia applications are exhaustive, causing network congestion to be a major problem. One way to deal with this problem is to use a resource reservation scheme, such as e.g. Tenet Scheme 2. This paper has proposed a security architecture for Tenet Scheme 2. The basic ideas are to use Internet layer security protocols, such as the IP Security Protocol (IPSP) and Internet Key Management Protocol (IKMP), to establish authentic communication channels between RCAP daemons, to handle client authentication and authorization locally, and to use a proxy-based mechanism to distribute access rights for target sets and channels. Note that none of these ideas is directly coupled to the internals of Tenet Scheme 2. It is thus assumed that the security architecture proposed in this paper is applicable to other resource reservation schemes, too. Also note that the security architecture uses as its building blocks a collision-resistant one-way hash function and a digital

signature system. Neither of them is subject to U.S. export controls. It is thus assumed that an implementation of the security architecture proposed in this paper would be exportable from the United States.

Acknowledgments

We would like to thank Domenico Ferrari for his encouragement and support through the design of the security architecture proposed in this paper.

References

1. R. Braden, L. Zhang, D. Estrin, S. Herzog, and S. Jamin. Resource ReServation Protocol (RSVP) — Version 1 Functional Specification. Internet Draft, November 1995. work in pogress.
2. F. Baker. RSVP Cryptographic Authentication. Internet Draft, November 1995. work in pogress.
3. D. Ferrari, A. Banerjea, and H. Zhang. Network support for multimedia – A discussion of the Tenet Approach. *Computer Networks and ISDN Systems*, 26:1267 – 1280, 1994.
4. A. Gupta and M. Moran. Channel Groups — A Unifying Abstraction for Specifying Inter-stream Relationships. TR-93-015, International Computer Science Institute (ICSI), Berkeley, CA, March 1993.
5. A. Gupta, W. Howe, M. Moran, and Q. Nguyen. Scalable resource reservation for multi-party real-time communication. TR-94-050, International Computer Science Institute (ICSI), Berkeley, CA, October 1994.
6. A. Gupta and D. Ferrari. Resource partitioning for multi-party real-time communication. TR-94-061, International Computer Science Institute (ICSI), Berkeley, CA, November 1994.
7. S. Deering and D.R. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2):85 – 110, 1990.
8. S. Deering. *Multicast Routing in a Datagram Internetwork*. PhD thesis, Stanford University, December 1991.
9. R. Braden, D. Clark, S. Crocker, and C. Huitema. Report of IAB Workshop on Security in the Internet Architecture, February 8-10, 1994. Request for Comments 1636, June 1994.
10. B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc., New York, NY, 1994.
11. D. Stinson. *Cryptography Theory and Practice*. CRC Press, Boca Raton, FL, 1995.
12. R.L. Rivest. The MD4 Message-Digest Algorithm. Request for Comments 1320, April 1992.
13. R.L. Rivest and S. Dusse. The MD5 Message-Digest Algorithm. Request for Comments 1321, April 1992.
14. NIST. Secure Hash Standard (SHS). FIPS PUB 180, Gaithersburg, MD, May 1993.
15. L. Gong. Using One-Way Functions for Authentication. *ACM Computer Communication Review*, 19(5):8 – 11, 1989.
16. G. Tsudik. Message Authentication with One-Way Hash Functions. *ACM Computer Communication Review*, 22(5):29 – 38, 1992.

17. R.L. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2):120 – 126, 1978.
18. T. ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithm. *IEEE Transactions on Information Theory*, IT-31(4):469 – 472, July 1985.
19. NIST. Digital Signature Standard (DSS). FIPS PUB 186, Gaithersburg, MD, May 1994.
20. ISO/IEC. Information Processing Systems — Open Systems Interconnection Reference Model — Part 2: Security Architecture. ISO/IEC 7498-2, 1989.
21. ISO/IEC. Information technology — Telecommunications and information exchange between systems — Network Layer Security Protocol. ISO/IEC 11577, 1993.
22. J. Ioannidis and M. Blaze. The Architecture and Implementation of Network-Layer Security Under Unix. In *Proceedings of the USENIX UNIX Security IV Symposium*, pages 29 – 39, Berkeley, CA, October 1993. USENIX Association.
23. NIST. Data Encryption Standard. FIPS PUB 46, Gaithersburg, MD, January 1977. Originally issued by National Bureau of Standards (NBS).
24. NIST. DES Modes of Operation. FIPS PUB 81, Gaithersburg, MD, December 1980. Originally issued by National Bureau of Standards (NBS).
25. P.C. Cheng, J.A. Garay, A. Herzberg, and H. Krawczyk. Design and Implementation of Modular Key Management Protocol and IP Secure Tunnel on AIX. In *Proceedings of the USENIX UNIX Security V Symposium*, Berkeley, CA, June 1995. USENIX Association.
26. A. Aziz, M. Patterson, and G Baehr. Simple Key-Management for Internet Protocols (SKIP). In *Proceedings of the Internet Society International Networking Conference*, June 1995.
27. H. Krawczyk. SKEME: A Versatile Secure Key Exchange Mechanism for Internet. In *Proceedings of the Internet Society Symposium on Network and Distributed System Security*, February 1996.
28. A. Liebl. Authentication in Distributed Systems: A Bibliography. *ACM Operating Systems Review*, 27(1):31 – 41, 1993.
29. B.C. Neuman. Proxy-Based Authorization and Accounting for Distributed Systems. In *Proceedings of the 11th International Conference on Distributed Computing Systems*, pages 283 – 291, May 1993.

This article was processed using the LaTeX macro package with LLNCS style