A Certificate Path Generation Algorithm for Authenticated Signaling in ATM Networks *

Jun Xu   Mukesh Singhal
Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210
{jun,singhal}@cis.ohio-state.edu

Abstract

ATM Forum specifies public key cryptography to be the default ATM authentication mechanism and directory services like X.509 to be the infrastructure for public key distribution and certification. Authenticated signaling, widely acknowledged as a necessary security feature of ATM network, requires the signaling message to be authenticated with a digital signature signed by the private key of the calling party. To verify the digital signature, the called party needs to obtain the public key of the calling party and a proof of the calling party's ownership to that public key. In X.509, the standard form of such a proof is a chain of public key certificates, called the certificate path between two parties. Certificate exchange protocol (CEP), proposed by ATM Forum, requires that another bi-directional connection be established between two parties to exchange public keys and certificate paths before an authenticated connection can be set up, which is not an ideal approach. We propose an algorithm which is embedded into ATM routing protocols to generate a certificate path inside a signaling message on-the-fly as the signaling message travels through the ATM network. In this approach, all that a calling party needs to know for authentication purpose is its own public key certificate and the ATM network builds the rest of the certificate path for it. Related issues like distribution of public key certificates and optimization of CA hierarchy are also addressed in this paper.

Keywords: ATM, P-NNI, authenticated signaling, certificate path.

*This work was partially supported by NSA Grant MDA904-96-1-0111.

1 Introduction

The initial design of TCP/IP gave little consideration to security issues. As a consequence, numerous security problems associated with the Internet (based upon TCP/IP) were identified [4], and adhoc solutions like firewalls were employed to patch these holes [3]. Unfortunately, none of these solutions is a panacea. In an effort to avoid the same pitfalls, ATM will be designed to provide security services at the network level, which offer an efficient and unified security mechanism to be used by applications running on ATM networks [14].

One of such services to be provided is the end-to-end authentication. ATM is a connection-oriented network, that is, a connection needs to be established between two parties (e.g., end systems) before they can talk to each other. The first type of connection, established by manually configuring the routing tables of the intermediate switches, is called *permanent virtual connection* (PVC). The second type of connection, known as *switched virtual connection* (SVC), is established automatically through a signaling protocol. Since much less network management and administration is required to set up and maintain SVC, it is more convenient and flexible than PVC, and thus more popular.

To establish a SVC, the calling party needs to send a *signaling message* SETUP to the called party. If the call can be accepted, the called party will send a signaling message CONNECT back to the calling party to confirm the connection [2]. To guard against threats like impersonation and spoofing, SETUP message should contain authentication information that the called party can use to verify the identity of the calling party. The same requirement applies to CONNECT message if mutual authentication is required. This problem is known as the *authenticated signaling* [14]. Since the problem and solution is symmetric for SETUP and CONNECT messages when mutual authentication is required, in the following discussion we will focus on SETUP signaling message only.

Public key cryptography (e.g., RSA) is chosen by ATM Forum to be the default authentication mechanism for ATM networks [14]. Authenticated signaling is done by inserting into the SETUP message a digital signature which is signed using the calling party's private key and is verified using the calling party's public key. For this scheme to work, the called party needs to obtain the public key of the calling party. To guard against *man-in-the-middle-attack*, the called party also needs to obtain a proof of the calling party's ownership to the public key, known as the *public key certificate* [11]. Public key certificates are issued to each party by a trusted third party called certification authority (CA) whose public key is known to every party [12]. A public key certificate contains the identity of the party, its public key, and some other information like its lifespan and serial number. It is signed by the private key of the CA so that no one can forge it. Each party can verify the validity of the certificate using the public key of the CA.

Due to scalability requirement, a large network usually employs multiple CAs, each of which issues certificates to a subset of nodes (e.g., switches, end systems, etc.). Based on the way how these CAs are organized, some CAs issue certificates to other CAs. For example, in X.509, all the CAs are organized in a hierarchy such that a higher level CA issues certificates to CAs one level below [12]. When a node A wants to talk to a node B,

A needs to obtain B's public key certificate. If B's certificate is signed by a CA C whose public key is not known to A, A needs to obtain the certificate of C as well. This process repeats until A obtains a certificate signed by a CA whose public key is known to A. Such a list of certificates that chains up trust relationships is called a *certificate path*.

Hence authenticated signaling requires the called party to obtain and verify the calling party's certificate path and vice versa. The ATM Forum proposes the *certificate exchange protocol* (CEP) for this purpose [14]. The basic idea behind CEP is to set up an unauthenticated bi-directional connection between the calling and the called party to exchange certificate path information. There are several drawbacks associated with this approach. First, the called party does not necessarily have the up-to-date information necessary for the calling party to compute a certificate path between them. In X.509 with hierarchical certification structure, such information is a certificate path from the root CA to the called party. It may keep changing due to periodical expiration and renewal of the certificates and to maintain a fresh copy of it at each party may not be a trivial task. Second, a CEP connection is needed before an authenticated connection can be established. If the called party does not have a fresh copy of the certificate path, additional connections are needed to query this information. So this approach incurs considerable overhead when the connection is being established. Finally, with this approach, it is hard to enforce the security policy "all connections should be authenticated" since CEP connections can not be authenticated. Such policy is useful for a secure ATM subnet, protected by a firewall switch which is equipped with a high-speed cryptography hardware and thus is able to authenticate all incoming SETUP signaling messages [13].

To overcome the drawbacks associated with CEP, we propose an algorithm, called *Certificate Path Generation Algorithm* (CPGA), that can be readily integrated into ATM routing protocols to generate a certificate path from the calling party to the called party on-the-fly as the signaling message travels through the network. In our approach, for authentication purpose, the calling party needs only include in the signaling message a digital signature and its public key certificate. While a signaling message is routed through the network, involved intermediate switches append to the message necessary certificates required to build the certificate path between the calling and the called party. When the message arrives at the called party, the certificate path has been built.

In the next section, we describe the network model to develop necessary background. In Section 3, we present our Certificate Path Generation Algorithm (CPGA) and dicuss its merits. Discussion of related issues is covered in Section 4. Section 5 concludes the paper.

2 The Network Model and Background

2.1 The Network Model

The network model of ATM networks under consideration is depicted in Fig. 1. This figure shows a large number of private ATM networks interconnected by a public ATM network,

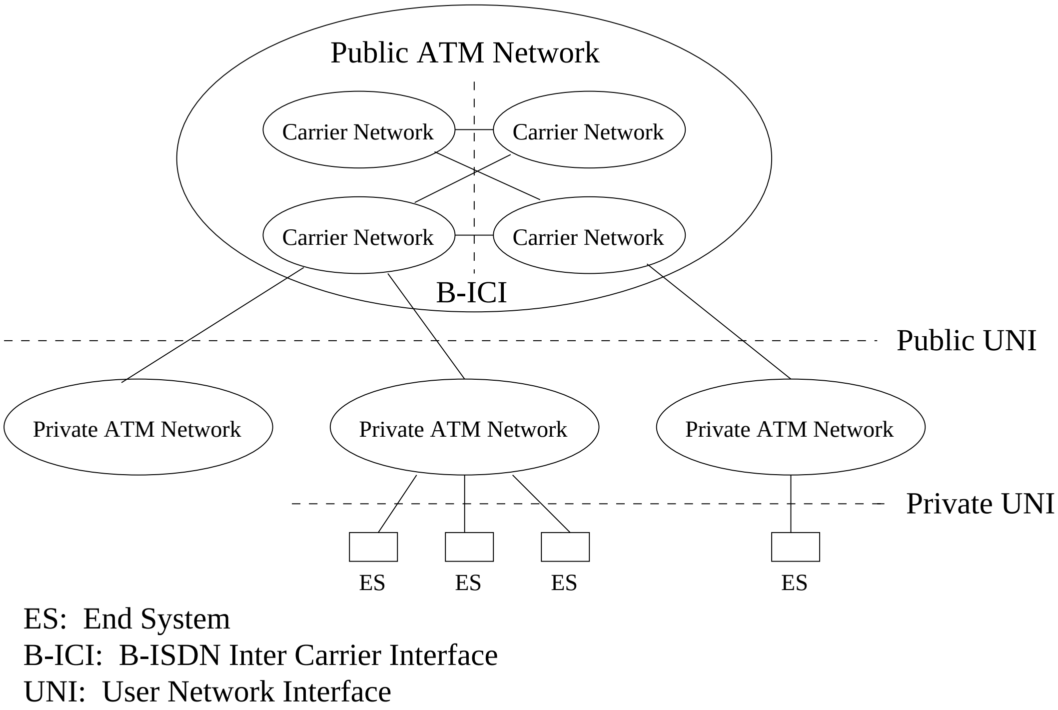

Figure 1: The network model.

which consists of a set of carrier networks connecting with each other through *BISDN Inter Carrier Interface* (B-ICI). Each private network has end systems attached to it, the number of which typically ranges from several to tens of thousands. If the calling and the called parties lie in the same private network, the signaling message is routed according to *Private Network-Network Interface* (P-NNI). Otherwise, the signaling message needs to be transported across the public network.

Since our certificate path generation algorithm is closely related to P-NNI, we give an overview of it next. A detailed tutorial on this topic can be found in [1]. Of course, the most complete and authoritative treatment of this subject is [5].

2.2 An Overview of P-NNI

2.2.1 Hierarchical Structure

P-NNI is a set of protocols used to route signaling messages in a private ATM network. P-NNI adopts a modified link-state algorithm, in which *P-NNI topology state packets* (PTSP) that carry topology and connectivity information, are distributed throughout the network. Based on such information, a switch decides how to forward a signaling message that arrives at it. If a flat structure is used when the network is large, the cost to distribute connectivity information between each pair of connected switches will be prohibitively high. For scalability reasons, P-NNI adopts a hierarchical structure such that switches are organized into groups and a higher level group consists of groups one level below. Only aggregate topology and connectivity information of a group is exchanged with other groups at the same level in the hierarchy [1].

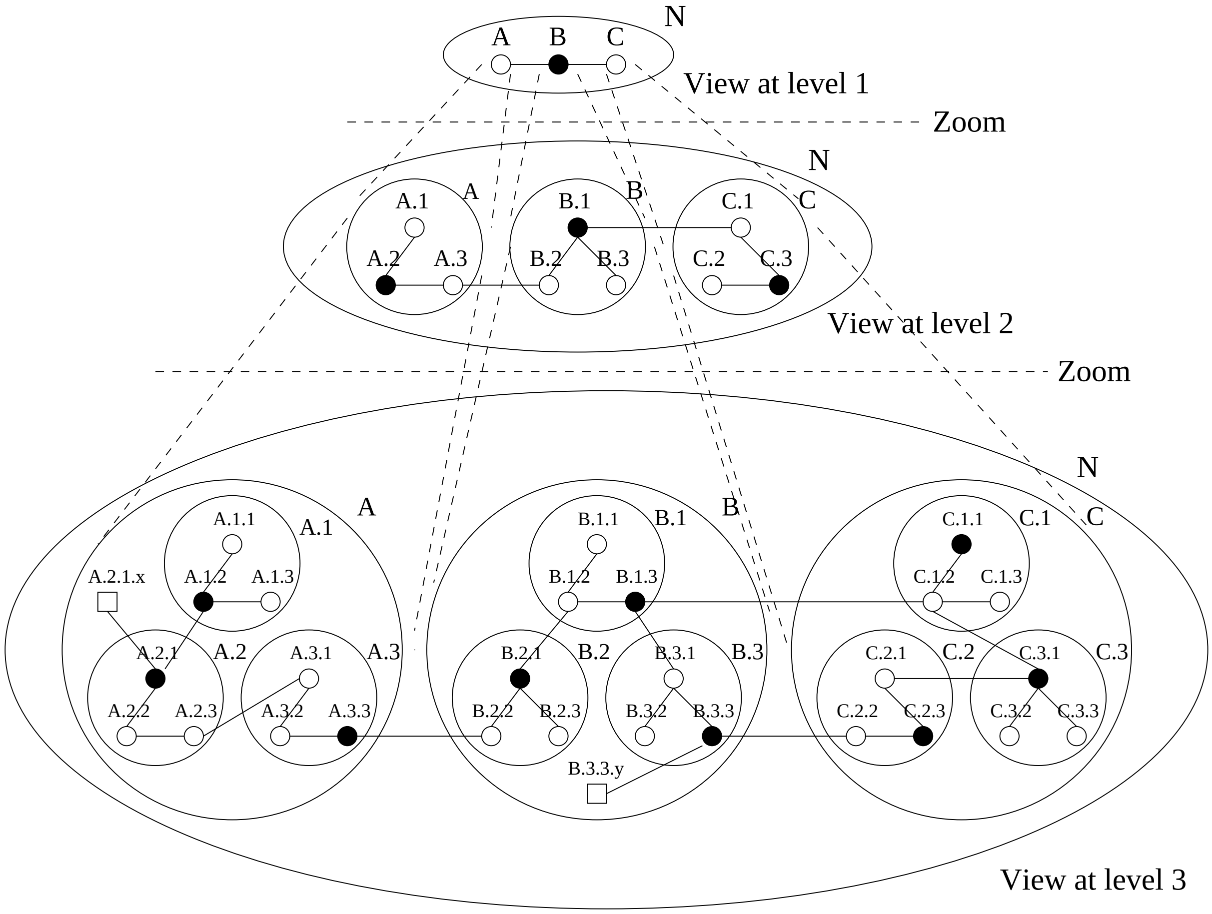

Figure 2: P-NNI hierarchical structure.

Fig. 2 shows an example of the hierarchical structure in a private ATM network. Viewed at level 1 (the part above the upper zoom line), the network N contains three high level groups (A, B and C), called *logical nodes*. Logical nodes are connected to each other through *logical links*. View at level 2 (the part between two zoom lines) zooms the view at level 1, and shows the detailed structure inside logical nodes A, B and C, and logical links between them. View at level 3 zooms the view at level 2, and shows the details of the network N down to the lowest level. At this level, each logical node is an ATM switch and each logical link is a physical link. For example, A.1.1, an ATM switch, is one of such lowest level logical nodes. End systems A.2.1.x is attached to switch A.2.1 and end system B.3.3.y is attached to switch B.3.3. A set of logical nodes at the same level are organized into a *peer group*. A peer group is abstracted into a *logical group node*

(LGN) at one level higher in the hierarchy. For example, in the view at level 3, A.1 is a peer group consisting of three logical nodes A.1.1, A.1.2 and A.1.3. While in the view at level 2, A.1 is a logical node in peer group A. A logical link may be one physical link or an abstraction of many physical links. For example, logical link B – C is actually composed of two physical links, B.1.3 – C.1.2 and B.3.3 – C.2.2. A logical node inside a peer group that has a logical link to a logical node in another peer group of the same level is called a *border node*. For example, A.2.3 is a border node at level 3 and A.3 is a border node at level 2. Such border nodes know that they have connections to other logical nodes and play an important role in routing [5].

Logical nodes in a peer group exchange full connectivity and topology information with each other. A logical node in a peer group only floods aggregate topology and connectivity information inside itself to other group members [1]. So, a node knows detailed information about the peer group (denoted by P, the parent) of which it is a member, less detailed information about the peer group (denoted by G, the grandparent) of which P is a member, even less detailed information about the peer group of which G is a member and so on. For example, the topology and connectivity information node A.2.1 has about the network N is shown in Fig. 3. The details of the mechanism to exchange such kind of information are out of the scope of this paper and can be found in [5]. Each peer group has a logical node that represents this peer group as a logical node in the parent peer group. This node is called a *peer group leader* (PGL). It summarizes the topology and connectivity information inside the peer group to be exchanged among the logical nodes of the parent group. In Fig. 2, each blackened logical node is a PGL.

2.2.2 Hierarchical Source Routing

Based on the knowledge a switch has about the whole network, P-NNI adopts a loose source routing technique, called *hierarchical source routing*. The source node gives a general direction about the route from the source to the destination. The direction does not include the complete route due to the limitation of the source's knowledge, but it is detailed enough so that intermediate nodes can follow the direction and route the signaling message accordingly. In P-NNI, such a direction is called a *designated transit list* (DTL).

We use the following example to explain the format and meaning of a DTL. If A.2.1 needs to send a signaling message to B.3.3.y, it includes in the message the DTL ([$\underline{\text{A.2.1}}$, A.2.2, A.2.3], [$\underline{\text{A.2}}$, A.3], [$\underline{\text{A}}$, B]). The DTL is a stack that contains three elements with the leftmost one as the top entry. The first element, which contains three items A.2.1, A.2.2, and A.2.3, specifies a path to route the message inside the peer group A.2. It reads "follow the corresponding physical links to go from A.2.1 to A.2.2 and then to A.2.3". Similarly, the second element reads "follow the logical link A.2 – A.3 inside A to travel from A.2 to A.3", and the last element reads "follow the logical link A – B inside N to travel from A to B". Given the information node A.2.1 has about the topology of the network N (shown in Fig. 3), the DTL is the best that A.2.1 can say about the route. Each element contains one and only one item that is underscored, and these items uniquely identify the position of the signaling message in the network. For example, when the DTL is initially created

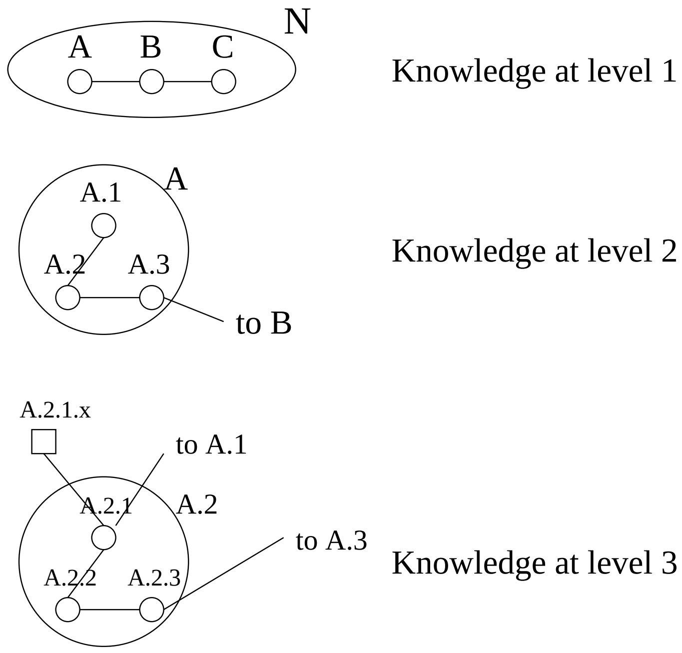

Figure 3: All the knowledge that A.2.1 has about the network.

at A.2.1, items A.2.1, A.2 and A are underscored denoting that the signaling message is now at A.2.1, which is inside A.2 and A.

A switch routes a signaling message according to the DTL included in the message. The top element specifies the route to follow when the message travels in the lowest level peer group the members of which are ATM switches. When the signaling message is about to leave a peer group for another, the node that the signaling message is visiting, which is a border node, pops it from the DTL because it is no longer useful. When the signaling message enters a new peer group, the node at which the signaling message arrives, also a border node, figures out how to route this message across this peer group (if the destination is outside the peer group) or to the destination (if the destination is inside this peer group). It creates a new element which specifies the route and pushes it to the DTL. When the message travels from one node to another, the pointer in the DTL, specified by the underscored items in each element, is updated accordingly.

We use the term *source switch* to refer to the switch that the calling party is attached to and *destination switch* the switch that the called party is attached to. The routing of a signaling message across the network consists of two stages. In the first stage, the signaling message first leaves the source switch. If the destination switch lies in the peer group (denoted by S_1) that contains source switch as one of its logical nodes, the first stage finishes. Otherwise, the message is routed outside S_1. If the destination switch does not lie in the parent peer group S_1 (denoted by S_2), the message is routed outside S_2. This process continues until the signaling message enters a peer group (denoted by S_n) that contains both the calling and the calling parties. At this moment, the first stage finishes and the second stage begins. In the second stage, if the destination switch is one of the logical nodes of S_n, the signaling message is routed to the destination switch according to the direction specified in the top element of the DTL. Otherwise, the signaling message is routed to the logical node of S_n that contains the destination switch (denoted by T_m) and enters it. If the destination switch is not one of the logical nodes of T_m, the signaling message is routed to the logical node of T_m that contains the destination switch. This process continues until the message is routed to the destination switch. At this moment the second stage finishes.

We explain the two stages by analyzing how a signaling message travels from switch A.2.1 to switch B.3.3 in Fig. 2. In the first stage, the signaling message first leaves A.2.1, the source switch, then peer group A.2, and finally peer group A. Both A.2 and A contain A.2.1. At this point, the signaling message enters peer group N, which contains both the source switch A.2.1 and the destination switch B.3.3. Then the message enters peer group B, then peer group B.3, and finally B.3.3, the destination switch. Both B and B.3 contain the destination switch B.3.3. Note that such a sequence is independent of the route the signaling message chooses to take.

2.3 Certificate Hierarchy

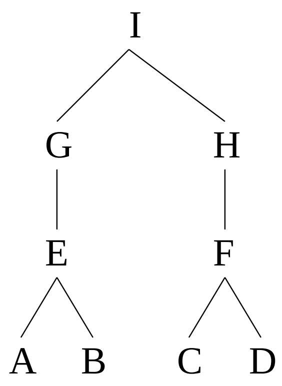

Figure 4: An example CA hierarchy.

ATM Forum specifies the X.509 directory service to be the default infrastructure for public key distribution and certification. Here we briefly introduce the basic idea and notations used in X.509 to make our discussion later easier. In X.509, CAs (Certification Authority) are organized in a tree-like hierarchy and Fig. 4 is an example of such a hierarchy [12]. Each leaf node is a principal (e.g., end systems, users) whose public key is to be certified and each tree node is a CA. All the child nodes of a tree node register their public keys with the CA that the tree node corresponds to. Each CA keeps a directory that contains the certificates of all the nodes registering their public keys with it and a certificate of the public key of its parent CA if it is not the root CA itself. The former are called *forward* *certificates* and the latter is called a *reverse certificate*. X.509 uses notation $X << Y >>$ to denote the certificate of Y issued by a CA X. In Fig. 4, E signs the certificates for A and B and its own certificate is signed by G. So E keeps forward certificates $E << A >>$ and $E << B >>$, and a reverse certificate $G << E >>$. $E << G >>$, $G << I >>$, $I << H >>$, $H << F >>$, and $F << C >>$ is the certificate path from C to A. This certificate path allows A to obtain and verify C's public key.

3 Certificate Path Generation Algorithm (CPGA)

Section 3.1 discusses the CA hierarchy used in our algorithm which is "isomorphic" to the P-NNI routing hierarchy. Section 3.2 shows how the algorithm constructs certificate paths by appending appropriate certificates to a signaling message as it passes through the network. Section 3.3 illustrates the operation of the algorithm by giving an example. Section 3.4 discusses the advantages of the algorithm over CEP. Section 3.5 discusses authenticated signaling when the calling and the called party are in different private ATM networks.

3.1 Organization of Certification Authorities

Since P-NNI comes with a nice routing hierarchy, our algorithm naturally fits into it by making the CA hierarchy "isomorphic" to it. We organize CAs in such a way that there is a CA in each peer group to certify each logical node in that group (certify its CA if it is a peer group or certify it if it is a single switch), and the root CA is the CA of the highest level peer group. At the lowest level, each switch functions as a CA that certifies all the end systems attached to it. For example, Fig. 5 shows the CA hierarchy for the routing hierarchy shown in Fig. 2. We use CA_P to denote the CA of a peer group P. In Fig. 5, the certificate path that A.2.1.x needs to provide to B.3.3.y for authentication purposes is $A.2.1 << A.2.1.x >>$, $CA_{A.2} << A.2.1 >>$, $CA_A << CA_{A.2} >>$, $CA_N << CA_A >>$, $CA_B << CA_N >>$, $CA_{B.3} << CA_B >>$, $CA_{B.3.3} << CA_{B.3} >>$.

3.2 The Algorithm

To establish an authenticated connection, the calling party first puts a signature of the message signed using its private key and its public key certificate into the signaling message. Then it sends the message to the source switch through User Network Interface (UNI).

The source switch appends its forward certificate to the signaling message and passes the signaling message to the next switch. When the signaling message leaves a peer group that contains the source switch, the border node of the peer group appends to the signaling message the forward certificate of the CA of the peer group. For example, in Fig. 2, if A.2.1.x sends a signaling message to B.3.3.y, nodes A.2, and A append

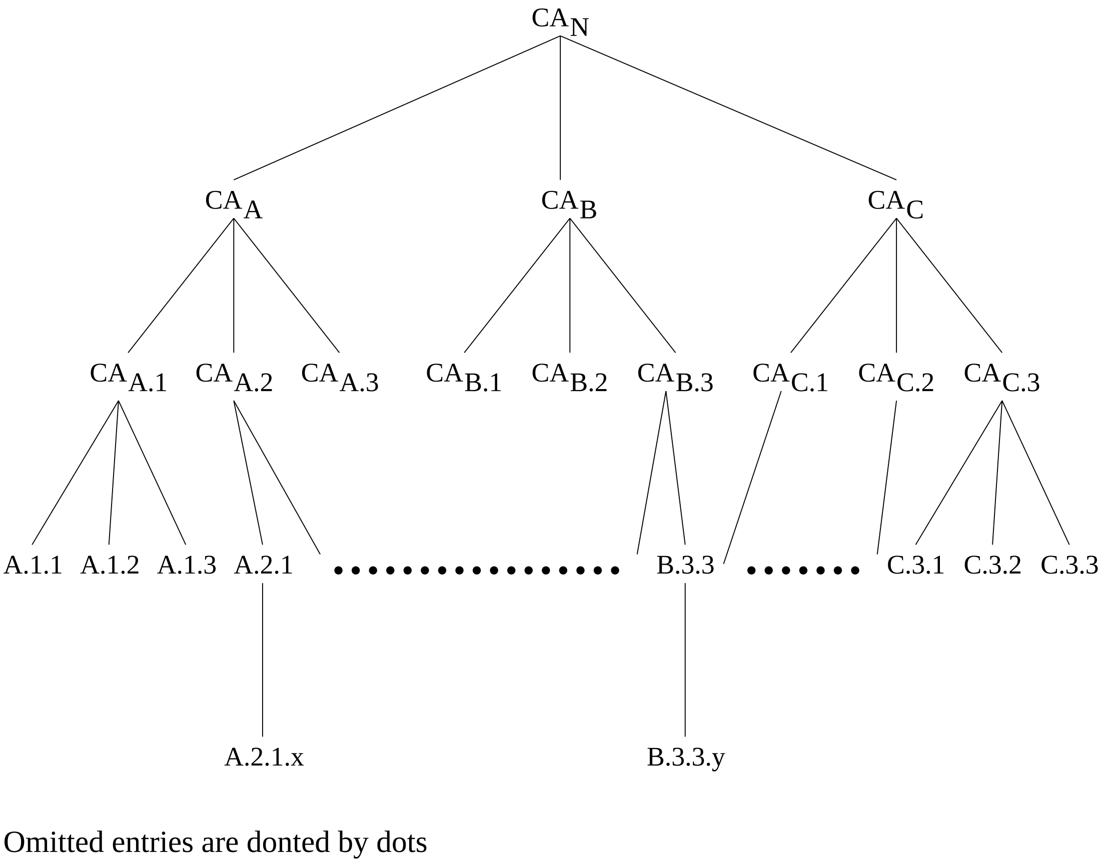

Figure 5: The CA hierarchy for network N.

certificates $CA_A << CA_{A.2} >>$ and $CA_N << CA_A >>$ to the message respectively when the message leaves the corresponding peer groups.

When the signaling message enters a peer group that contains the destination switch, the border node of the peer group appends to the signaling message the reverse certificate of the CA of the peer group. Continuing with the example of Fig. 2, nodes B and B.3 append certificates $CA_B << CA_N >>$ and $CA_{B.3} << CA_B >>$ to the signaling message respectively when the message enters the corresponding peer groups. When the destination switch receives the signaling message, it appends its reverse certificate the signaling message.

When the signaling message arrives at the called party, the signature and the certificates appended to the signaling message look like $E_A(Msg)$, $S_1 << A >>$, $S_2 << S_1 >>$, ..., $S_n << S_{n-1} >>$, $T_m << S_n >>$, $T_{m-1} << T_m >>$, ..., $T_1 << T_2 >>$. A is the calling party and $E_A(Msg)$ is the signature of the message signed with A's private key. S_1 is the source switch, and S_2, S_3, ..., and S_{n-1} are the CAs of the peer groups the signaling message is leaving in the first stage. Each S_i is a logical node of the peer group S_{i+1}, $i = 1, 2, ..., n-1$. S_n is the peer group that contains both the source and destination switches. T_{m-1}, T_{m-2}, ..., and T_2 are the peer groups that the signaling message is entering in the second stage, and T_1 is the destination switch. Each T_i is a logical node of the peer group T_{i+1}, $i = 1, 2, ..., m - 2$ and T_{m-1} is a logical node of the peer group S_n.

When the signaling message arrives at the called party, it is authenticated as follows. Since the called party knows the public key of T_1, it verifies the certificate $T_1 << T_2 >>$ to obtain the public key of T_2. Using the public key of T_2, it verifies the certificate $T_2 << T_3 >>$ to obtain the public key of T_3 and so on. In the end, it obtains the public key of A from the certificate $S_1 << A >>$ and uses it to verify if it is the calling party A who signed the signature $E_A(Msg)$. Once this is done the signaling message is authenticated.

In order to append appropriate certificates to the signaling message when the message leaves or enters a peer group, the border nodes of the peer group have a copy of the forward certificate issued to its CA from the CA of its parent peer group. The border nodes also keep a copy of the corresponding reverse certificate. For example, in Fig. 2, node A.2.3, a border node of A.2, keeps the certificates $CA_A << CA_{A.2} >>$ and $CA_{A.2} << CA_A >>$. How border nodes obtain fresh copies of such certificates will be discussed in Section 4.2.

The pseudocode of the algorithm is shown in CSP like syntax as follows:

Algorithm CPGA
begin
<the signaling message is leaving the source switch> =>
"append the forward certificate of the source switch to the signaling message."
□
<the signaling message is leaving a peer group that contains the source switch> =>
"append the forward certificate of the CA of the peer group to the signaling message."
□
<the signaling message is entering a peer group that contains the destination switch> =>
"append the reverse certificate of the CA of the peer group to the signaling message."
□
<the signaling message arrives at the destination switch> =>
"append the reverse certificate of the destination switch to the signaling message."
end

The algorithm is executed at each switch that a signaling message traverses. The DTL contained in the signaling message is used to determine whether the message is leaving a peer group that contains the source switch (the second condition in the above pseudo code) or is entering a peer group that contains the destination switch (the third condition in the above pseudo code). When a switch is about to pop the top element from the DTL and all other elements in the DTL have their first items underscored, this means the message is leaving a peer group that contains the source switch. When a switch is about to push a new element into the DTL and all other elements in the DTL have their last items underscored, this means the message is entering a peer group that contains the destination switch.

3.3 An Example

We now illustrate the algorithm with the help of an example. We use the network topology shown in Fig. 2. Following the path that a signaling message takes from A.2.1.x to B.3.3.y, we show security-related operations at intermediate switches.

The calling party A.2.1.x signs the message using its private key, appends the signature and its public key certificate $A.2.1 << A.2.1.x >>$ to the signaling message, and then signals A.2.1 through UNI [7].

When A.2.1 receives the signaling message, it appends the certificate $CA_{A.2} << A.2.1 >>$ to the signaling message because the signaling message is about to leave the source switch. Then it sends the message to A.2.2.

When A.2.2 receives the signaling message, none of the four conditions in the algorithm is met; therefore A.2.2 does not perform any security-related operations and simply forwards the message to A.2.3.

When A.2.3 receives the signaling message, it finds out that the message is about to leave the peer group A.2, which contains the source switch A.2.1. Hence the second condition in CPGA is met and the switch appends $CA_A << CA_{A.2} >>$, the forward certificate of the CA of A.2, to the signaling message. It then sends the message to A.3.1.

The border node of A.3, namely, A.3.1, receives the signaling message. Switch A.3.1 finds out that nothing needs to be done because the peer group that the message is entering is A.3, which does not include the destination switch, B.3.3. So it simply forwards the message to A.3.3.

When A.3.3 receives the message, it finds out that the message is about to leave the peer group A, which contains the source switch. Hence the second condition in CPGA is met and A.3.3 appends the forward certificate of the CA of A, $CA_N << CA_A >>$, to the signaling message. It then sends the message to B.2.2.

When B.2.2 receives the signaling message, it figures out that the signaling message is entering the peer group B, which contains the destination switch B.3.3. Hence the third condition in CPGA is met and the switch appends the reverse certificate of the CA of B, $CA_B << CA_N >>$, to the signaling message. It then sends the message to B.2.1.

When B.2.1 receives the signaling message, since none of the four conditions in the algorithm is met, B.2.1 does not perform any security related operations. It simply forwards the message to B.1.2, which finds out that no security-related operations need to be done and simply forwards the message to B.1.3. B.1.3 finds out that no security-related operations need to be done and simply forwards the message to B.3.1.

When B.3.1 receives the signaling message, it finds out that the message is entering the peer group B.3, which contains the destination switch B.3.3. Hence the third condition in CPGA is met and the switch appends the reverse certificate of the CA of B.3, $CA_{B.3} << CA_B >>$, to the signaling message. B.3.1 then sends the message to B.3.3.

B.3.3 finds out that it is the destination switch; so it appends its reverse certificate, $B.3.3 << CA_{B.3} >>$, to the signaling message and sends the signaling message to B.3.3.y through UNI.

Upon receipt of the signaling message, B.3.3.y extracts from it the complete certificate path from A.2.1.x to B.3.3.y consisting of the certificates appended at the intermediate switches. Since it knows the public key of B.3.3, it can verify the certificate $B.3.3 << CA_{B.3} >>$ and obtain the public key of $CA_{B.3}$. It uses the public key of $CA_{B.3}$ to verify the certificate $CA_{B.3} << CA_B >>$ and obtain the public key of CA_B, and so on. This process continues until B.3.3.y obtains the public key of A.2.1.x, and uses it to verify the signature signed using the private key of A.2.1.x.

If the signature is proved to be made by A.2.1.x, the signaling message is authenticated. When mutual authentication is required, the CONNECT message sent from B.3.3.y to A.2.1.x will be processed in the same way as the SETUP message.

3.4 Advantages over CEP

The algorithm has several advantages over CEP, which are listed as follows:

- First, all the information an end system needs to know for authenticated signaling is reduced to a minimum, its public key certificate. This greatly reduces the complexity of the network software at end systems.

- Second, no CEP connection is needed before an authenticated connection can be established. So the response time for an authenticated signaling is reduced at least by a factor of two.

- Third, since all signaling messages are authenticated, we can enforce the security policy "all connections must be authenticated", which is useful in implementing the ATM firewalls [1].

3.5 Authenticated Signaling across Public ATM Networks

The certificate path generation algorithm (CGPA) we discussed in Section 3.2 works when both the calling and the called parties are in the same private ATM network. When two end systems are attached to different private networks, the signaling message between them must be carried across a public network. In this section, we show how this algorithm can be augmented to "bridge" the certificate path across a public network.

In Fig. 6, two private networks N1 and N2 are connected to the public network through public *User Network Interface* (UNI) I1 and I2, respectively. The border switch in private network N1 that is directly attached to the public network is S1, and S2 in private network N2. End system ES1 is attached to network N1 and it intends to establish a connection with end system ES2, which is attached to network N2. Both N1 and N2 use the CA hierarchy and CPGA algorithm we proposed in Section 3.2. The root CAs of N1 and N2

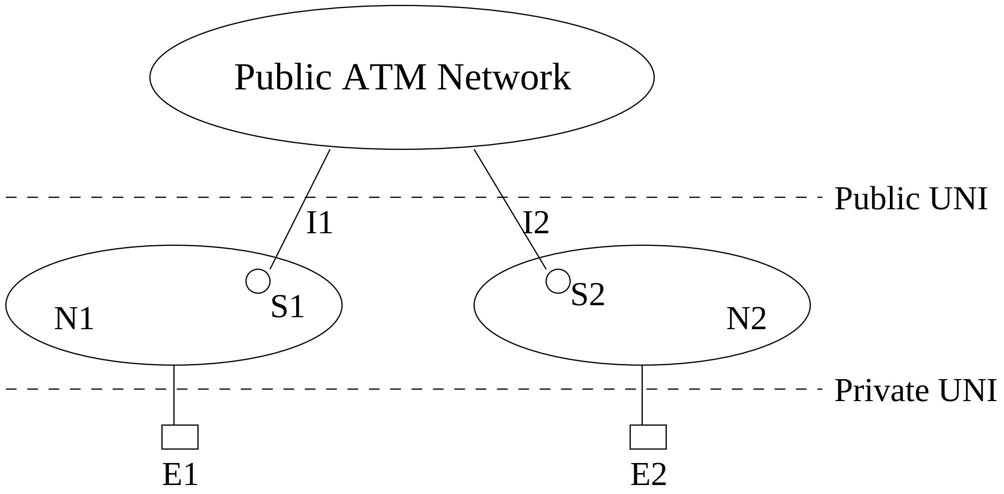

Figure 6: Authenticated signaling across a public network.

are CA_{N1} and CA_{N2}, respectively. In the following, we describe two possible approaches to route signaling message from ES1 to ES2 and discuss how to build a certificate path across the public network in both cases.

The first approach is called *permanent virtual path tunneling* [1]. In this approach a *permanent virtual path* (PVP) is set up between N1 and N2, more specifically between I1 and I2, across the public network. Signaling request from ES1 is first routed to border switch S1 of network N1, then tunneled within the PVP through the public network to border switch S2 of network N2, and finally routed to ES2. Using CPGA, if N1 and N2 are considered peer logical nodes connected by a logical link (the PVP), when the signaling message is routed from ES1 to S1, the certificate path from ES1 to CA_{N1} is built and when the message is routed from S2 to ES2, the certificate path from CA_{N2} to ES2 is built. What is missing from the certificate path between ES1 and ES2 is a certificate path from CA_{N1} to CA_{N2}. Our solution to this problem is to let CA_{N2} issue the certificate $CA_{N2} << CA_{N1} >>$ to CA_{N1} and vice versa. When the signaling message is routed to S1 and is about to be tunneled through the public network, S1 will append to signaling message the certificate $CA_{N2} << CA_{N1} >>$, which glues up the certificate paths built in N1 and N2. Therefore, all N1 and N2 need to do for authenticated interconnection is to let the root CAs certify each other when the PVP is set up.

The other approach works if the public network supports *switched virtual connection* (SVC) through public UNI [1]. In this approach, the P-NNI signaling request from the switch that end system ES1 is attached to is mapped to a public UNI signaling request at border switch S1 and the public ATM network will route this signaling request all the way to border switch S2, map it back to a P-NNI signaling request and route it to the switch that end system ES2 is attached to [1]. This approach is preferred since it allows more flexibility and ubiquitous communication needs. But up to now SVC signaling is not provided in public ATM networks because B-ICI between the carrier networks does not support any control plane functions [6]. However, since future version of B-ICI is expected to support SVC signaling across it, we assume that SVC support will be available soon [8]. Under this assumption, the alternative approach works as follows. Each public UNI between a private network and the public network is assigned a unique E.164 address. E.164 addressing scheme was originally used in telephone networks and now used in public ATM networks. It is different from NSAP, the addressing scheme used in private ATM networks. When the signaling message from ES1 is routed to S1, S1 will map it into a signaling message from I1 (the public UNI between network N1 and the public network) to I2 (the public UNI between network N2 and the public network) that uses the E.164 address of I2 as the destination. When the signaling message is routed to I2, S2 will map it back to a signaling message destined for ES2. The public ATM network is expected to provide a directory service to enable S1 query the E.164 address of I2 according to the NSAP address of ES2 [1]. When a private network is connected to the public network, it registers its NSAP address with the public network and is given the E.164 of the public UNI it is attached to. So public networks know the unique binding between a NSAP address of a private network and its E.164 address. Our algorithm will be enhanced in such a way that each private network registers the public key of its root CA with the public network and the public network will use a hierarchical CA structure. The directory service is enhanced in such a way that when S1 queries for the E.164 address of I2, the certificate path from I1 to I2 will be provided to S1 as well. Our algorithm is enhanced in such a way that S1 will append the certificate path from I1 to I2 queried from the directory service described above to the signaling message. In this way the pieces of certificate paths built in N1 and N2 get glued up at S1 to form a complete path certificate.

4 Related Issues

4.1 Dedicated CA Logical Nodes

We assumed that there is one CA in each peer group and we may let the PGL nodes play the role of CAs. However for ease of administration and management, we can introduce a special logical node, called *dedicated CA logical node* into each peer group to act as the CA of that peer group. Each dedicated CA logical node consists only of one switch. This switch does not route any user traffic; instead it is dedicated to perform the functions of a CA for that peer group. It signs and distributes the public key certificates of fellow logical nodes in the peer group. Below we examine the functions that can be provided efficiently by such dedicated CA logical nodes.

4.2 Distribution of Public Key Certificates

In X.509, when a public key certificate is revoked before it expires, each party has to be notified about it and each party has to keep a list of all the revoked certificates. If we shorten the lifespan of public key certificates to several minutes, the need to explicitly revoke a certificate is generally eliminated. In our approach, the CA node simply does not renew the certificate that has been revoked. For this scheme to work, there are two requirements. First, the clocks among the switches and end systems should be synchronized. There are many ways to synchronize the clocks in a distributed system and they can be found in [9, 10]. Second, the CA should be able to distribute the certificates to each party periodically. With the help of the dedicated CA logical node, the distribution of certificates is straightforward. Since PTSP (P-NNI topology state packets) is flooded periodically inside the peer group, we modify its format to include a number of slots for storing public key certificates. The PTSP packets sent from the CA logical node will fill in these slots renewed public key certificates (signed by the CA logical node) of the fellow logical nodes in the peer group. Since the packet will be flooded inside the peer group, each logical node can pick up its own certificate. We assume that PTSP packets are exchanged every L (secs.), the lifespan for the certificate of each party is a constant I (secs.), and there are N slots for certificates in a PTSP packet. In a peer group consisting of P logical nodes, we only need to make sure that $L > P/N * I$ so that the certificate of each party is renewed promptly. In this way, the public key distribution is seamlessly built into the P-NNI.

When a PTSP packet containing the certificate of the peer group (as a node in parent group) enters the peer group, the border node that receives the PTSP will first keep a copy of the certificate and then flood it within the peer group to make sure that every other border node gets a copy. In such a way each border node in a peer group maintains a valid copy of the certificate of the peer group (as a logical node in the parent peer group).

4.3 A Variant of CA Hierarchy

On some occasions, there may be only a small number of logical nodes in a peer group. In these situation, we can remove CAs from some levels and "compress" the hierarchy. For example, we can compress the CA hierarchy shown in Fig. 5 by removing CAs from A.1, A.2, ..., C.3 peer groups and the resulting CA hierarchy is shown in Fig. 7. The modification to the algorithm is that the border nodes will append a certificate if one of the four conditions is met and the peer group the signaling message is entering or leaving has its own CA node. For example, in Fig. 7, there is no CA node that serves A.1 and all logical nodes in A.1 peer group are served by the CA of peer group A CA_A. This optimization shows that the CA hierarchy need not be "isomorphic" with the P-NNI routing hierarchy and it only needs to be "compatible".

5 Conclusion

Authenticated signaling in ATM networks requires that the called party obtains a certificate path between the calling party and the called party. We examined that although

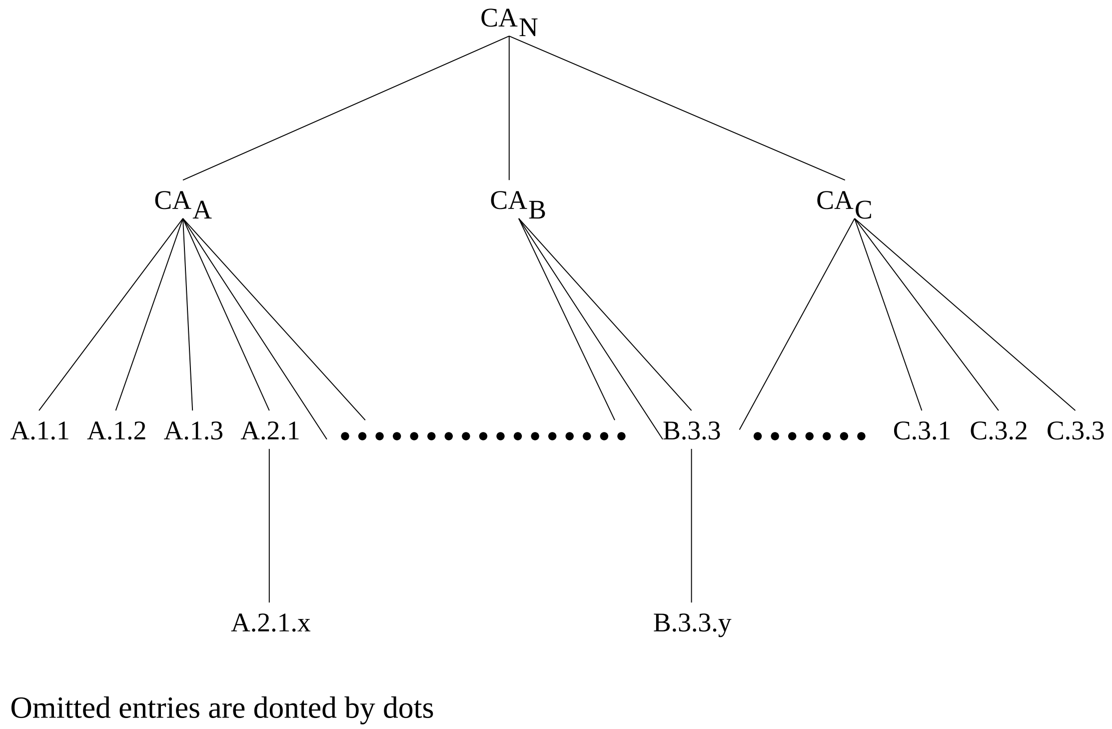

Figure 7: A "Compressed" version of CA hierarchy in network N.

CEP (Certificate Exchange Protocol) offers a solution to the problem, it suffers from certain drawbacks like complexity of maintaining certificate paths at each end system, high latency in establishing an authenticated connection, etc. In this paper, we proposed an algorithm to generate the certificate path in a signaling message on-the-fly while the signaling message travels from the calling party to the called party. Compared to CEP, the proposed algorithm has three advantages. First, all the information an end system needs to know for authenticated signaling is reduced to a minimum, its public key certificate. This greatly reduces the complexity of the network software on end systems. Second, no CEP connection is needed before an authenticated connection can be established. So the response time for an authenticated signaling is reduced at least by a factor of two. Third, since all signaling messages are authenticated, we can enforce the security policy "all connections must be authenticated", which is useful in implementing the ATM firewalls. Though our algorithm is mainly based on the P-NNI routing hierarchy, it is not restricted to private ATM networks. In Section 3.5 we examined two possible ways of interworking with public ATM networks and show how to extend our algorithm to work in each case. Finally we discussed the related issues like the distribution of certificates and the optimization of CA hierarchy.

References

[1] A. Alles, “ATM Internetworking”, Internet posting, Cisco Systems, Inc., http://www.cisco.com/warp/public/614/12.html, May 1995.

[2] U. Black, ATM: Foundations for Broadband Networks, Prentice Hall PTR, Englewood Cliffs, NJ, 1995.

[3] D. B. Chapman, and E. D. Zwicky, Building Internet Firewalls, O’Reilly & Associates, Sebastopol, CA, 1995.

[4] W. R. Cheswick, and S. M. Bellovin, Firewalls and Internet Security, Addison Wesley, Reading, MA, 1994.

[5] ATM Forum – ”Private Network-Network Interface Specification Version 1.0 (PNNI 1.0)”, March 1996.

[6] ATM Forum – ”BISDN Inter Carrier Interface (B-ICI) Specification Version 2.0”, December 1995.

[7] ATM Forum – ”ATM User-Network Interface (UNI) Version 3.1”, September 1994.

[8] ATM Forum – ”Addendum to BISDN Inter Carrier Interface (B-ICI) Specification, Version 2.0 (B-ICI Specification, Version 2.1)”, September 1996.

[9] K. Marzullo, and S. Owicki, ”Maintaining the Time in a Distributed System”, *ACM Proceedings of the Second Annual ACM Symposium on Principles of Distributed Computing*, pp 295-305, Montreal, Quebec, CA, August 1983.

[10] D. L. Mills, “Improved Algorithms for Synchronizing Computer Network Clocks”, *IEEE/ACM Transactions on Networking*, vol. 3, pp. 245-254, June 1995.

[11] B. Schneier, Applied Cryptography (2nd ed.), John Wiley & Sons, Inc., NY, 1996.

[12] W. Stalling, Network and internetwork security: principles and practice, Prentice Hall, Englewood Cliffs, NJ, 1995.

[13] D. Stevenson, N. Hillery, and G. Byrd, ”Secure Communications in ATM Networks”, *Communications of ACM*, vol. 38, pp. 45-52, February 1995.

[14] T. D. Tarman – Editor, Sandia National Laboratories, ”Phase I ATM Security Specification”, ATM Forum/95-1473R5, October 1996.