# Certificate Path Generating Protocol (CPGP) for Authenticated Signaling in ATM Networks

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

*Abstract*— **Authenticated signaling is an important security service to be provided by ATM networks to guard against threats of spoofing and impersonation. 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. With public key cryptography, authenticated signaling 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. CEP (Certificate Exchange Protocol), 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 established, which is not an ideal approach. We propose a Certificate Path Generating Protocol (CPGP), which is embedded into ATM signaling and routing protocols to generate a certificate path inside a signaling message on-the-fly as the signaling message travels through the ATM network. In CPGP, all that a calling party needs to do for authenticated signaling is to put into the signaling message its own public key certificate and the digital signature of the signaling message signed using its private key. CPGP builds the rest of the certificate path for it. The proposed protocol is nicely embedded into the ATM signaling and routing protocol so that no performance overhead is incurred to establish the certificate path.**

## I. Introduction

*Authenticated signaling* is an important security service to be provided by ATM networks to guard against threats of spoofing and impersonation [1]. It also serves as a support service for other security services such as *session key exchange* [1]. In *authenticated signaling*, the *SETUP* signaling message needs to contain the authentication information of the calling party that is verifiable by the called party. The same requirement applies to *CONNECT* message if mutual authentication is required. Since the problem and solution is symmetric for *SETUP* and *CONNECT* messages when mutual authentication is required, only one direction is considered in following discussion. Public key cryptography (e.g., RSA) is chosen by ATM Forum to be the default authentication mechanism for ATM networks [1]. With public key cryptography, the authentication information contained in the SETUP message is a *digital signature* signed using the calling party's private key. In order to verify the signature, 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* [2], [3].

In a small network, public key certificates are issued to each party by a trusted third party called *certification authority* (CA), whose public key is universally known [4], [3]. A public key certificate issued to a party contains the identity of the party, the public key, and the lifespan of the certificate. The certificate is signed using the private key of the CA and can be verified using its public key. For scalability, a large network needs to employ multiple CAs, each of which serves a part of the network. There should be a structure among those CAs so that parties served by different CAs can authenticate each other. X.509 is chosen as the infrastructure for public key distribution and certification for ATM networks. In X.509, CAs are organized into a tree structure [4]. Each leaf

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

node is a party (e.g., an end system) and each tree node is a CA. Each CA issues *forward certificates* to its children and a *reverse certificate* to its parent CA. In X.509, notation $X << Y >>$ denotes the public key certificate issued to $Y$ by a CA $X$. If a party $A$ wants to verify the *public key certificate* of another party $B$, signed by a CA $C$, $A$ needs to obtain the certificate of $C$ if $A$ does not know $C$'s public key. 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*. For example, in the CA hierarchy shown in Fig. 1, the *certificate path* from C to A, which allows A to obtain and verify C's public key, consists of a *reverse certificate* $E << G >>$, and *forward certificates* $G << F >>$ and $F << C >>$. This *certificate path* is verifed by A as follows. Knowing the public key of E, A uses it to verity $E << G >>$ to obtain the public key of G. Then, A uses the public key of G to verity $G << F >>$ to obtain the public key of F. Finally, A uses the public key of F to verify $F << C >>$ to obtain the public key of C.

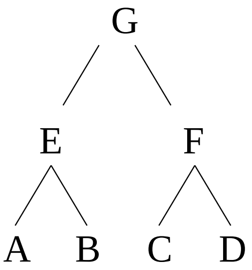


Fig. 1. An example CA hierarchy.

Hence *authenticated signaling* requires the called party to obtain and verify the calling party's *certificate path* and vice versa. The ATM Forum proposes CEP (Certificate Exchange Protocol) for this purpose [1]. 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 calling party does not necessarily have the up-to-date information necessary for the called party to compute the *certificate path* between them. In X.509 CA hierarchy, such information is a *certificate path* from the calling party to the root CA. It may keep changing due to periodical expiration and renewal of the certificates and to maintain a fresh copy of it at each party is not 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 delay 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 are not authenticated. This security policy is useful for securing an ATM subnet using a firewall switch, equipped with a high-speed cryptography hardware to authenticate all incoming SETUP signaling messages [5].

To overcome the drawbacks associated with CEP, we propose CPGP (Certificate Path Generating Protocol), which generates a *certificate path* from the calling party to the called party on-the-fly as the signaling message travels through an ATM network. All that the source party needs to do for *authenticated signaling* is to put into the signaling message its public key certificate and the digital signature of the signaling message. While the signaling message is routed through the network, involved intermediate switches will insert into the signaling message the certificates required to build the *certificate path*, and a complete *certificate path* is inside the signaling message when the message arrives at the called party. The proposed protocol is nicely embedded into the ATM signaling and routing protocol so that no performance overhead is incurred to establish the *certificate path*.

The proposed protocol works in both a private ATM network and across a public ATM network. In the next section, we present how CPGP works in a private ATM network. In Section 3, we show how CPGP works across a public ATM network. Section 4 concludes the paper.

## II. CPGP in a Private ATM Network

### A. Background of P-NNI

A private ATM network employs *Private Network-Network Interface* (P-NNI) as its routing and signaling protocol [6]. PNNI employs a hierarchical topology, in which switches are grouped into *peer groups* and a higher level *peer group* consists of a number of lower level *peer groups* as its *logical nodes*. There can be a *logical link* between two *logical nodes*, which is the abstraction of one or more underlying physical links. In a *peer group*, a *logical node* that has a *logical link* to another peer group is called a *border node* of the peer group. *Border nodes* play an important role in PNNI routing and signaling as well as in executing CPGP. Fig. 2 shows the hierarchical topology in a

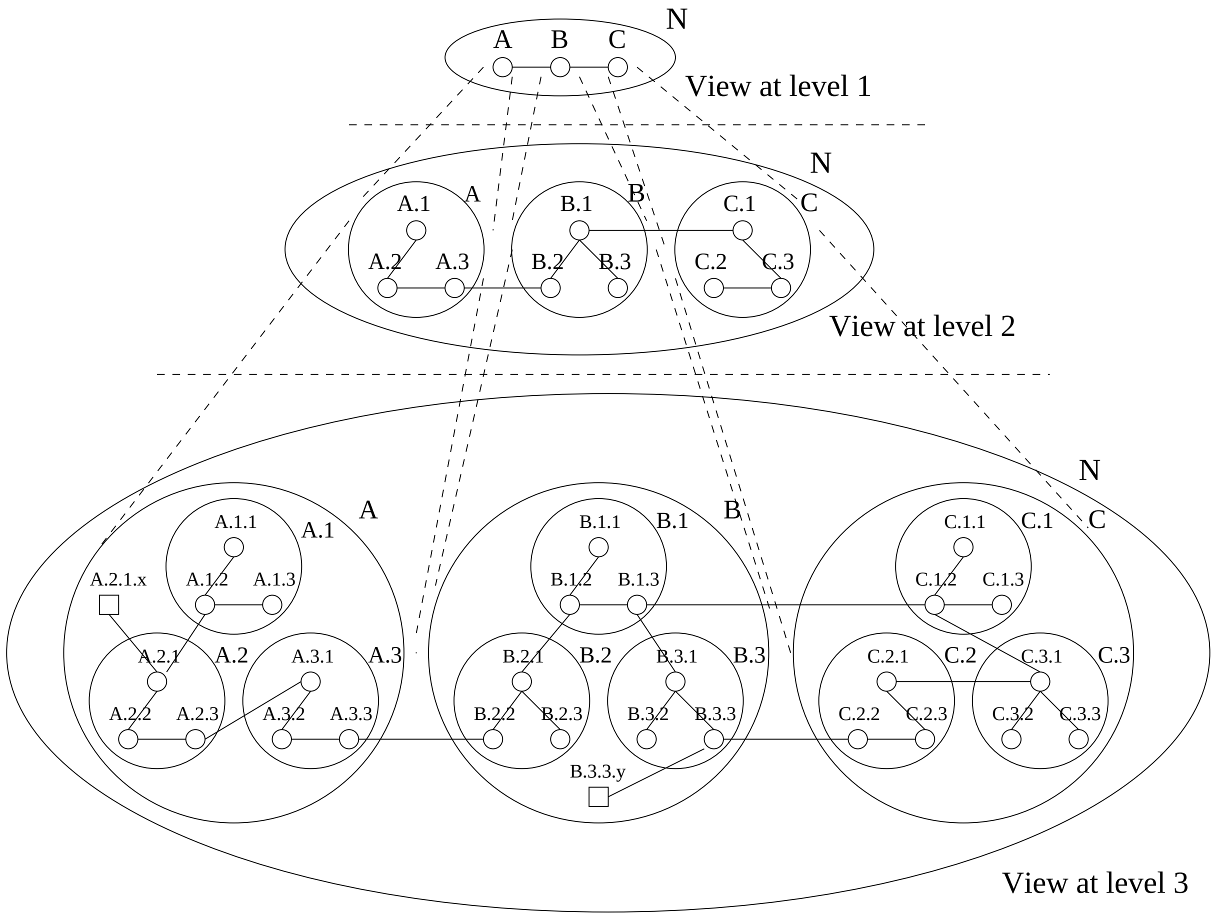


Fig. 2. P-NNI hierarchical structure

private ATM network. View at level 1 shows that the network N contains three logical nodes, namely, A, B, and C. View at level 2 zooms the view at level 1, which shows that A is itself a *peer group* that contains three lower level *logical nodes*, namely, A.1, A.2, and A.3. View at level 3 illustrates 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. End systems A.2.1.x and B.3.3.y are attached to switch A.2.1 and switch B.3.3, respectively.

We can view the routing of a signaling message from an *ingress switch*, to which the source end system is attached, to the *egress switch*, to which the destination end system is attached, as consisting of two stages. In the first stage, the signaling message first leaves the ingress switch, then the *peer group* that contains the ingress switch as a *logical node* (denoted as $S_1$), then the parent *peer group* of $S_1$ (denoted as $S_2$), ..., until the signaling message leaves a peer group (denoted as $S_{n-1}$) the parent of which (denoted as $S_n$) contains both the *ingress switch* and the *egress switch*. At this moment, the first stage finishes and the second stage begins. In the second stage, the signaling message is first routed to the logical node of $S_n$ that contains the egress switch (denoted by $T_m$) and enters it, then enters the logical node of $T_m$ that contains the egress switch, ..., until the message reaches the egress switch. For example, in Fig. 2, a signaling message from switch A.2.1 (on behalf of end system A.2.1.x) to switch B.3.3 (on behalf of end system B.3.3.y) consists of following steps. In the first stage, the message first leaves ingress switch A.2.1, then leaves peer group A.2, and finally leaves peer group A. In the second stage, the message first enters the peer group B, then B.3, and finally reaches B.3.3.

*Hierarchical source routing* is used in PNNI to route the signaling message from the *ingress switch* to the *egress switch*. In the signaling message, the ingress switch specifies a *hierarchically complete* route as to how the signaling message should be forwarded, encoded in a stack-based data structure called *designated transit list* (DTL). DTL is interpreted and manipulated by the intermediate switches in order to determine the "next hop" to forward the signaling message [7]. The detail of DTL and the operations to be performed on it at intermediate switches are outside the scope of this paper.

### *B. The Proposed Protocol*

We assume that when a signaling message arrives at the ingress switch through *user network interface* (UNI) [8], it contains the *public key certificate* of the source end system and the *digital signature* of the signaling message. The objective of the protocol is to construct the rest of the *certificate path* from the source end system to the destination end system.

CPGP in a private ATM network consists of two parts. The first part is to configure the X.509 CA hierarchy of the network to make it "isomorphic" to the P-NNI routing hierarchy of the network. The second part is the protocol to be executed at certain intermediate switches in the network when the signaling message travels through them.

#### B.1 Configuration of X.509 CA Hierarchy

CPGP arranges the CA hierarchy of a private ATM network to be "isomorphic" to the PNNI routing hierarchy as follows. There is a CA in each peer group to certify the CA of each logical node in the peer group. CA of a lowest level *peer group* (consists of switches) certifies all the end systems attached to the switches that belong to the peer group. For example, Fig. 3 shows the CA hierarchy for the routing hierarchy shown in Fig. 2, in which $CA_P$ denotes the CA of a peer group P. The border nodes of each peer group keeps the *forward certificate* issued to the CA of the peer group from the CA of the parent peer group. The border nodes also keep the *reverse certificate* issued from the CA of the peer group to the CA of the parent peer group. For example, in Fig. 2, *A.2.3*, a *border node* of A.2, keeps the *forward certificate* $CA_A << CA_{A.2} >>$ and the *reverse certificate* $CA_{A.2} << CA_A >>$. How border nodes obtain up-to-date certificates will be discussed in Section 2.5.2.

Independently, in its PNNI 2.0 specification, ATM Forum proposed a guideline for establishing a certification hierarchy in a PNNI network, which requires the PNNI certification hierarchy to be "functionally identical" to PNNI routing hierarchy [9]. This certification hierarchy is used in PNNI 2.0 for authentciation among *logical nodes* of a *peer group* to securely exchange topology and reachability information. This certification hierarchy, when instantiated, will be equivalent to our CA hierarchy used in CPGP. We proposed our CA hierarchy for authenticated signaling earlier than PNNI 2.0 [10].

#### B.2 Protocol Execution

The protocol consists of the following two actions executed at the *border nodes* of each peer group that contains either the *ingress switch* or the *egress switch*:

(I) When the signaling message is about to leave a *peer group* that contains the *ingress switch*, the *border node* of the *peer group* should insert the *forward certificate* issued to the CA of the peer group from the CA of the parent peer group. (This operation is nicely embedded into *DTL processing* in a switch as follows. When the switch is about to pop the top *entry* from the *DTL stack*, this indicates that the signaling message is about to leave a *peer group*. If the *pointer* in each other *entry* of DTL points to the first *item* inside the *entry*, this indicates that the *peer group* contains the *ingress switch* [7].)

(II) When the signaling message is about to enter a *peer group* that contains the *egress switch*, the *border node* of the *peer group* should insert the *reverse certificate* issued from the CA of the peer group to the CA of its parent peer group. (This operation is nicely embedded into *DTL processing* of a switch as follows. When the switch is going to push a new *entry* to the DTL, this indicates that the signaling message is entering a *peer group*. If the *pointer* in each other *entry* of DTL points to the last *item* inside the *entry*, this indicates that the *peer group* contains the *egress switch* [7].)

We now illustrate the protocol with an example. We use the network topology shown in Fig. 2. The end system A.2.1.x wants to establish an authenticated SVC with the end system B.3.3.y. So, it puts its public key certificate $A.2 << A.2.1.x >>$ into the signaling message and sends it to A.2.1. Suppose the signaling message follows the route: A.2.1, A.2.2, A.2.3, A.3.1, A.3.2, A.3.3, B.2.2, B.2.1, B.1.2, B.1.3, B.3.1, and B.3.3. The protocol is executed as follows:

- When the signaling message arrives at switch A.2.3, switch A.2.3 finds out that the message is about to leave the peer group A.2, which contains the ingress switch A.2.1. According to (I), the *forward certificate* $CA_A << CA_{A.2} >>$ is inserted into the signaling message.
- When the signaling message arrives at switch A.3.3, switch A.3.3 finds out that the message is about to leave the peer group A, which contains the ingress switch A.2.1. According to (I), the

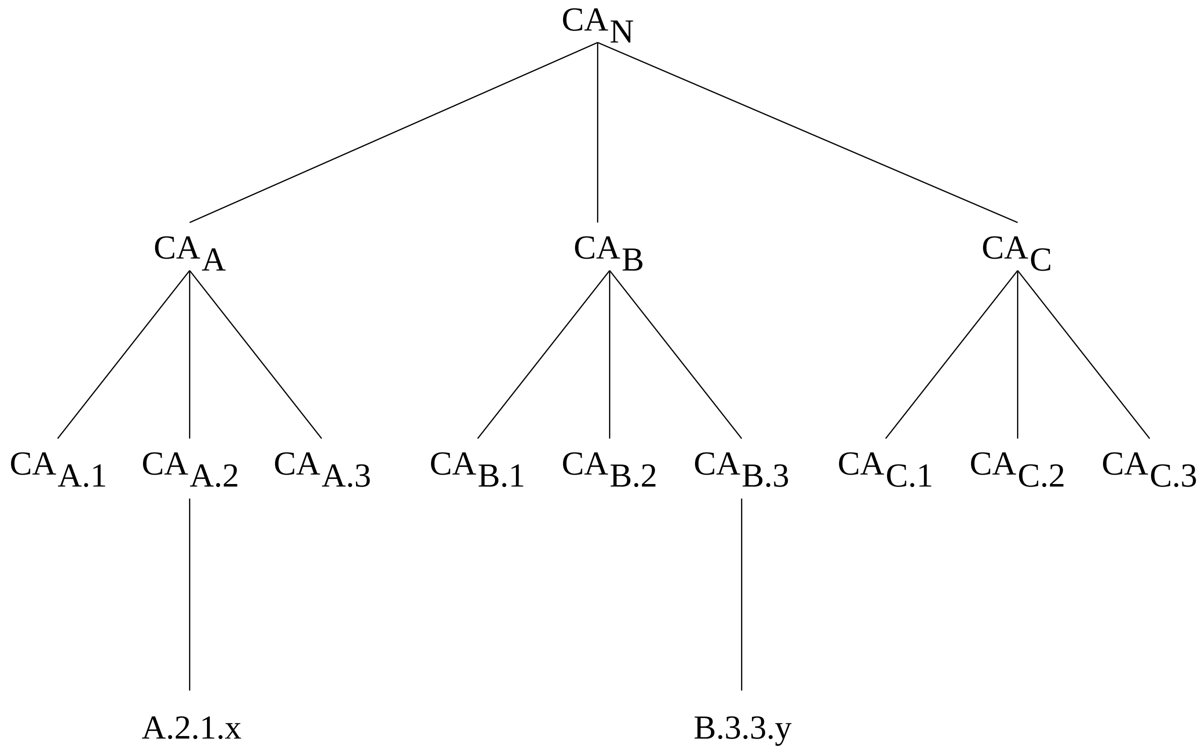


Fig. 3. The CA hierarchy for network N.

*forward certificate* $CA_N << CA_A >>$ is inserted into the signaling message.

- When the signaling message arrives at switch B.2.2, switch B.2.2 finds out that the signaling message is entering the peer group B, which contains the egress switch B.3.3. According to (II), the *reverse certificate* $CA_B << CA_N >>$ is inserted into the signaling message.
- When the signaling message arrives at switch B.3.1, switch B.3.1 finds out that the signaling message is entering the peer group B.3, which contains the egress switch B.3.3. According to (II), the *reverse certificate* $CA_{B.3} << CA_B >>$ is inserted into the signaling message.

When the signaling message reaches B.3.3, above four steps have been executed. As a consequence, two *forward certificates*, $CA_A << CA_{A.2} >>$ and $CA_N << CA_A >>$, and two *reverse certificates*, $CA_B << CA_N >>$ and $CA_{B.3} << CA_B >>$, are inserted into the signaling message. We can see that these four certificates, together with $A.2 << A.2.1.x >>$ (the public key certificate of A.2.1.x), compose a complete *certificate path* from A.2.1.x to B.3.3.y.

### C. Correctness Argument

**Theorem**: In a private ATM network that deploys CPGP, a signaling message sent from an end system A to another end system B will contain the *certificate path* from A to B.

**Proof**: For any end system E, let $E_1$ denote the switch to which E is attached, and $E_{i+1}$ denote the parent peer group of $E_i$ (for i = 1, 2, ...). Suppose the lowest level *peer group* in the network that contains both the ingress and egress switch is denoted as $A_m = B_n$, which means that it is the (m-1)th grandparent *peer group* of the ingress switch and is also the (n-1)th grandparent *peer group* of the egress switch. As discussed in Section 2.1, in the first stage of the traversal, the signaling message first leaves *peer group* $A_2$, then *peer group* $A_3$, ...., until it leaves *peer group* $A_{m-1}$. According to (I), the *forward certificates* inserted into the signaling message by the *border nodes* of these *peer groups* are $A_3 << A_2 >>$, $A_4 << A_3 >>$ ..., and $A_m << A_{m-1} >>$. In the second stage of the traversal, the signaling message first enters *peer group* $B_{n-1}$, then *peer group* $B_{n-2}$, ..., until it reaches the egress switch $B_1$. According to (II), the *reverse certificates* inserted into the signaling message by the *border nodes* of these *peer groups* are $B_{n-1} << A_m >>$ ($A_m$=$B_n$), $B_{n-2} << B_{n-1} >>$, ..., and $B_2 << B_3 >>$. The source end system A will also put its public key certificate $A_2 << A >>$ into the signaling message. When the signaling message reaches the destination end system, it will contain certificates $A_2 << A >>$, $A_3 << A_2 >>$, $A_4 << A_3 >>$, ..., $A_m << A_{m-1} >>$, $B_{n-1} << A_m >>$, $B_{n-2} << B_{n-1} >>$, ..., and $B_2 << B_3 >>$, which is the complete certificate from A to B ($B_2$'s public key is known to B since it serves as B's CA).

### D. Processing Overhead and Advantages Over CEP

In terms of processing overhead, the protocol execution at a switch consists of two parts. The first part is to examine whether the signaling message is leaving a *peer group* that contains the *ingress switch* or is entering a *peer group* that contains the *egress switch*. As explained in the protocol, this part is embedded into the relatively time-consuming DTL processing and its overhead is absorbed. The second part is to insert a *certificate* into the signaling message. The overhead of this part is also negligible if the insertion position is known in advance (e.g., maintain a pointer at the message header) or the certificate is appended at the end.

CPGP 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 [7].

### E. Related Issues

#### E.1 Dedicated CA Logical Nodes

We assumed that there is one CA in each peer group and we may let the *peer group leader* (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* in each peer group to act as the CA of that peer group. Each dedicated CA logical node is a host or device that is devoted to performing 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.

#### E.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 *dedicated CA logical 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 [11], [12]. Second, the *dedicated CA logical node* should be able to distribute the certificates to each party periodically. With the dedicated CA logical node, the distribution of certificates is straightforward. Since *PNNI topology state packets* (PTSP) are 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 and the *reverse certificate* of the CA of the parent *peer group*. Since the packet will be flooded inside the peer group, each logical node can pick up its own certificate. In addition, the *border nodes* will also pick up the *reverse certificate* of the CA of the parent peer group. 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 $L > (P+1)/N * I$ so that the certificate of each party is renewed promptly (P *forward certificates* and one *reverse certificate*). In this way, the key distribution is seamlessly built into the PNNI.

When a PTSP packet containing the *forward 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 *forward certificate* of CA of the peer group.

### F. Practical Considerations

In CEP, caching of public key certificates at the source is an effective way of reducing the number of CEP executions. Our protocol also benefits from the scheme. In CPGP, if the calling party has already obtained all the necessary certificates for building the certificate path, it will indicate in the signaling message that CPGP needs not be invoked in the intermediate switches.

In X.509, a public key certificate can be as long as 900 bytes. When the certificate path between two parties is long, the certificates will be a nontrivial portion of the signaling message payload. A hybrid protocol of CEP and CPGP can be used to solve this problem. When the certificate path is long, CEP is used to exchange the certificates out-of-band. When the certificate path is short, CPGP is used to generate the certificate path in-band.

## III. CPGP Across a Public ATM Network

CPGP across a public ATM network is an extension of CPGP in a private ATM network. Consider the network topology shown in Fig. 5. Two private ATM networks, N1 and N2, are connected to the public network through public *user network interfaces* (UNI), I1 and I2, respectively. S1 and S2 are *border switches* that are directly attached to the public ATM network in N1 and N2, respectively. End system ES1 is attached to N1 and it intends to establish a connection with end system ES2, which is attached to N2. Signaling message from ES1 is first routed to border switch S1, then transported across the public network through an interworking protocol, to border switch S2, and from there routed to ES2.

CPGP across a public ATM network is summarized as follows. Private ATM networks N1 and N2 deploy CPGP as described in the last section. The root CAs of N1 and N2 are $CA_{N1}$ and $CA_{N2}$, respectively. N1 and N2, connected by a *logical link* (the public ATM network), are considered *logical nodes* of the same "virtual peer group". According to CPGP in a private network, 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. Now, what is needed for building the certificate path from ES1 to ES2 is a certificate path from $CA_{N1}$ to $CA_{N2}$. There are two possible interworking approaches to transport signaling message from ES1 to ES2 across the ATM network. How CPGP builds the certificate path from N1 to N2 in each interworking approach is presented next.

(I) The first interworking approach is called *permanent virtual path tunneling* [7]. In this approach, a *permanent virtual path* (PVP) is set up between S1 and S2. Signaling request from ES1 is first routed to border switch S1, then tunneled within the PVP through the public network to border switch S2, and from there routed to ES2. In this interworking approach, when the PVP is set up, CPGP lets $CA_{N2}$ issue the certificate $CA_{N2} << CA_{N1} >>$ to $CA_{N1}$ and vice versa (cross-certify each other). 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.

(II) The other approach is to let public ATM networks support signaling through public UNI [7]. In this approach, the PNNI signaling message originated from the ingress switch is mapped to a public UNI/NNI signaling message at border switch S1. The public ATM network will route the public NNI signaling message all the way to border switch S2 using its own routing protocol. At S2, the public NNI signaling message is mapped back to a PNNI signaling message and routed it to the egress switch. One of the mapping functions performed at the public UNI is address mapping; the addressing scheme used in public ATM networks is E.164 (the addressing scheme originally used in telephone networks), while in private ATM networks is NSAP. The public ATM network is expected to provide a directory service to enable S1 query the E.164 address of I2, which is needed to route the signaling message across the public ATM network, according to the NSAP address of ES2 [7]. In this interworking approach, CPGP works as follows. When a private ATM network subscribes a public UNI from the public ATM network, it also registers the public key of its root CA with the public network. There is a CA hierarchy inside the public network (may not be X.509) so that there is a certificate path between the root CAs of any two subscribers. The directory service of the public ATM network is enhanced in such a way that when S1 queries for the E.164 address of I2, the certificate path from $CA << N1 >>$ to $CA << N2 >>$ will be provided to S1 as well.

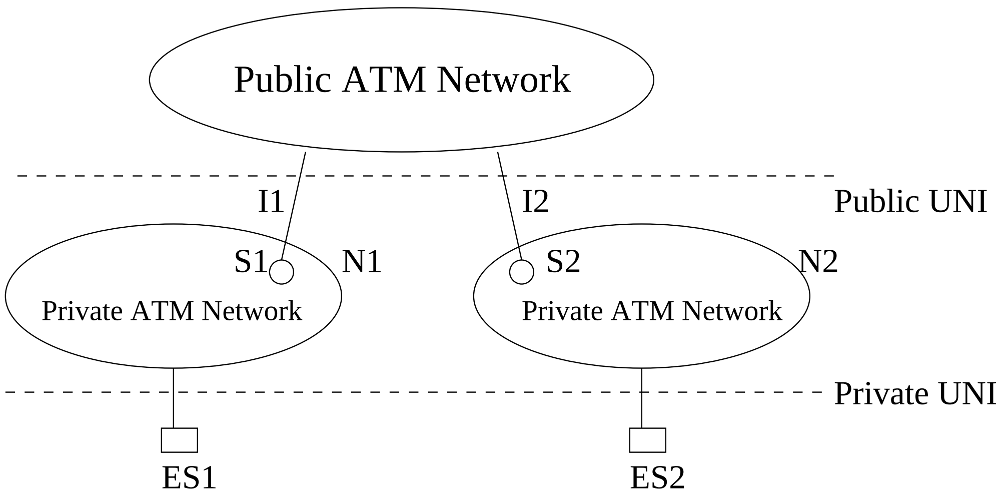


Fig. 4. Authenticated signaling across a public network.

## IV. Conclusion

*Authenticated signaling* in ATM networks requires that the called party obtains a *certificate path* from the calling party to the called party and vice versa. We examined that although 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 a protocol 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. The proposed protocol is nicely embedded into the ATM signaling and routing protocol so that no performance overhead is incurred to establish the *certificate path*. The proposed protocol works in both a private ATM network as well as across a public ATM network. Compared to CEP, the proposed protocol has three advantages. First, all that 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 ATM firewalls.

## References

[1] T. Tarman, "Phase i atm security specification," Tech. Rep., ATM Forum, Oct. 1996.

[2] B. Schneier, *Applied Cryptography*, John Wiley & Sons, NY, 2 edition, 1996.

[3] T. Woo and S. Lam, "Authentication for distributed systems," *IEEE Computer*, pp. 39–52, Jan. 1992.

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

[5] D. Stevenson, N. Hillery, and G. Byrd, "Secure communications in atm networks," *Communications of ACM*, pp. 45–52, Feb. 1995.

[6] ATM Forum, *Private Network-Network Interface (PNNI) Specification Version 1.0*, Mar. 1996.

[7] A. Alles, "Atm internetworking," Tech. Rep., Cisco Inc., May 1995.

[8] ATM Forum, *ATM User-Network Interface (UNI) Version 3.1*, Sept. 1994.

[9] ATM Forum, *Private Network-Network Interface (PNNI) Specification Version 1.0*, Sept. 1997.

[10] J. Xu and M. Singhal, "A certificate path generation algorithm for authenticated signaling in atm networks," Tech. Rep., The Ohio State University, May 1997.

[11] K. Marzullo and S. Owicki, "Maintaining the time in a distributed system," in *2nd Annual ACM Symposium on Principles of Distributed Computing*, Aug. 1983, pp. 295–305.

[12] D. Mills, "Improved algorithms for synchronizing computer network clocks," *IEEE/ACM Transactions on Networking*, pp. 245–254, June 1995.