Securing QoS: Threats to RSVP Messages and their Countermeasures[*]

Tsung-Li Wu, S. Felix Wu
Zhi Fu, He Huang

Computer Science Department
North Carolina State University
Raleigh, NC 27695
wu@csc.ncsu.edu

Feng-Min Gong

Advance Networking Research
MCNC
RTP, NC 27709-2889
gong@mcnc.org

December 18, 1998

Abstract

In this paper, we study the first type of *DoQoNS (Denial of Quality of Network Service)* attacks: **attacks directly on the resource reservation and setup protocol**. In particular, we have studied and analyzed the RSVP protocol. Two important research contributions are presented: First, we performed a security analysis on RSVP which demonstrates the key vulnerabilities of its distributed resource reservation and setup process. Second, we proposed a new secure RSVP protocol, *SDS/CD (Selective Digital Signature with Conflict Detection)* for RSVP, which combines the strength of *attack prevention* and *intrusion detection*. SDS/CD resolved a fundamental issue in network security: how to protect the integrity, in an End-to-End fashion, of a target object that is mutable along the route path. As a result, we will show that SDS/CD can deal with many *insider* attacks that can not be handled by the current IETF/RSVP security solution: *hop-by-hop Authentication*.

[*] This research is supported by the U.S. Department of Defense Advanced Research Projects Agency under contract DABT63-97-C-0045.

1 Introduction

Traditional "best-effort" service provided by Internet was not designed to support specified, desired, and consistent levels of quality of service (QoS) of network traffic. Modern network applications such as distributed multimedia, IP-telephony and video conferencing are very sensitive to the QoS parameters such as end-to-end latency, bandwidth, and packet loss rate. In order to support the service requirements for these new applications, frameworks and protocols have been proposed to provide different levels of network service assurance. Within IETF, for example, new frameworks/protocols are being developed and standardized under a few working groups including Differential Services (DiffServ), ReSerVation setup Protocol (RSVP/RAP), Multiple Protocol Label Switching (MPLS), and Quality of Service Routing (QoSR).

On one hand, in the near future, one or more advanced network service frameworks and protocols will be standardized and implemented by the network and telephony industry. We expect that the new Internet services will be securer, more flexible and predictable. On the other hand, these new network services themselves become the new targets for denial of service attacks. We call these attacks “Denial of Quality of Network Service (DoQoNS)” attacks.

Attacks to quality of network services (DoQoNS) can be classified into two different types: control flow attacks (*e.g.*, killer reservation in RSVP [Tal98]) and data flow attacks (e.g., resource stealing in RSVP/DiffServ). First, an intruder can attack directly on the signaling/control protocol for network resource reservation and connection setup. For example, as we will discuss later, a vicious insider can interfere with the RSVP signaling protocol to deny the service of a particular link. The result of such an attack will make this link conceptually unavailable to other users.

Second, even if the resource reservation process is successful, the intruder can still attack the dataflow such some or all the reserved resources are not spent appropriately. For instance, in DiffServ [BBB+98, BBC+98], if an attacker can “cheat” or “by-pass” the first router (i.e., a local router), it will inject K false “Expedited Forwarding (EF)” packets, all with the same destination IP address. Now, we assume that another legitimate user in the same domain reserved $2K$ but only sent K EF packets. Therefore, the “Ingress” router of a neighbor domain will allow these K false EF packets going through because the aggregated flow ($K_{good} + K_{false}$) is under the $2K$ limit. At a later point, these two flows ($K_{good} + K_{false}$) of packets will splitted by a router because their destination addresses are different. Right after the split, these K_{false} packets start consuming the resources reserved for other legitimate packet flows, which will likely degrade the QoS significantly for other flows.

In this paper, we focus ONLY on the first type of *DoQoNS (Denial of Quality of Network Service)* attacks: **attacks directly on the resource reservation and setup protocol**. In particular, we have studied and analyzed the RSVP protocol. Two important research contributions are presented: First, we performed a security analysis on RSVP which demonstrates the key vulnerabilities of its distributed resource reservation and setup process. Second, we proposed a new secure RSVP protocol, *SDS/CD (Selective Digital Signature with Conflict Detection)* for RSVP, which combines the strength of *attack prevention* and *intrusion detection*. We will demonstrate the security strength and practicality of this new protocol. And, we will also show that SDS/CD can deal with many *insider* attacks that can not be handled by the current IETF/RSVP security solution: *hop-by-hop authentication* [BLT98].

2 QoS and RSVP

Since a few years ago, Internet Engineering Task Force (IETF) has proposed integrated services model, which includes the *Control Load (CL)* service [Wro97a], *Guaranteed (G)* classes [SPG97], and *Best Effort (BE)* service, to define QoS along the network. The Guaranteed service guarantees that datagrams will arrive within the guaranteed delivery time and will not be discarded due to queue overflows, while the Controlled-Load service is defined to imply a commitment to that provided to best-effort traffic under lightly loaded conditions.

A resource reservation set-up protocol known as *RSVP (ReSerVation protocol)* [ZBE+97] is proposed under IETF to support a variety of QoS control services including CL and G classes by working with Integrated Service. Under the RSVP framework, *a service sender* (*e.g.,* a video provider) sends out special *path finding* messages ($PATH(sender \leftarrow receiver)$) periodically toward the service receiver (if unicast) or all potential receivers (if multicast). These PATH messages and the data packet flow are delivered to the receiver(s) through the same routing path(s). PATH messages not only implicitly discover the routing path(s) but also collect the QoS information along the path(s). For instance, the sender's Tspec object (Tspec($PATH$)) specifies the traffic characteristics, while the Adspec object (Adspec($PATH$)) represents the minimum resource level available for the routing path. On the other hand, a receiver will periodically send the reservation messages ($RESV(receiver \leftarrow sender)$) along the reversed routing path toward the sender. The Flowspec object (Flow_spec($RESV$)) in the $RESV(receiver \leftarrow sender)$messages specifies the desired QoS, while the Filterspec object (Filter_spec($RESV$)), together with a session specification, defines the set of data packets – the "flow" – to receive the QoS defined by Flow_spec($RESV$). Some of the RSVP objects remain unchanged from sender to receiver (or from receiver to sender) under normal operation (e.g., Tspec($PATH$), Filter_spec($RESV$), or Flow_spec($RESV$)under the unicast case), and others may be updated by the intermediate RSVP routers (e.g., Adspec($PATH$), or Flow_spec($RESV$)under the multicast case).

All RSVP-enabled routers along the route path from sender to receiver create a corresponding data structure being called *Path State Block (PSB)* for each session being identified by *destination IP address, port, and protocol identifier*. The *Previous Hop (PHOP)* object in the PATH messages is used to identify the RSVP-enabled entities downstream on the path.

Upon receiving the first PATH message from sender for a new session, the receiver makes decision whether to "order" this session of service. If the receiver decides to accept the offer, a RESV message is sent out toward the sender along the reversed path. This RESV message carries the information about QoS level requested by the receiver for this session. Upon receiving RESV messages, an intermediate RSVP-enabled router on the path makes a decision whether to grant the reservation request according to their own policies [YPG98]. Also, a corresponding data structure called *Reservation State Blocks (RSBs)* is created for each new session. Also, the reservation styles could be either *Wildcard-Filter (WF)* style, *Fixed-Filter (FF)* style, and *Shared-Explicit (SE)* style (for performing a multicast-merge to allow different reservations sharing resources). A successful reservation for a new session will not happen unless all of RSVP-enabled routers along the path grant the reservation requests.

To keep the reservation valid dynamically, the succeeding RESV messages will be sent out toward the sender periodically to refresh the corresponding RSBs along the path [ZBE+97]. The state blocks in RSVP-enabled routers are deleted if no matching refresh PATH or RESV messages arrive before the *"cleanup time-out"* interval. State blocks may also be deleted immediately by an explicit "TearDown" message, *i.e.,* a PATH or RESV TearDown message. A TearDown request may be initiated by an application in a sender, receiver, or an RSVP router as the results of either state block time-out or service preemption. There are two time parameters relevant to reservation state: the refresh period R between two PATH/RESV refresh messages and the local state lifetime L. The value of L is determined by the value of R which is specified in PATH/RESV message. In [ZBE+97], the current suggested value for R is 30 seconds and at least about 157.5 seconds for L (*i.e.*, roughly missing 5 refresh messages).

In RSVP, route change is possible during a session. The local routing protocol can notify the RSVP process of route change for particular destinations. The corresponding PATH/RESV refresh message for those destinations will re-create PSBs/RSBs respectively along the new path.

3 Attacker's Objectives and Classification for RSVP

3.1 Attacker's Objectives

The main focus of our study is on RSVP QoS attacks. We consider four objectives for the QoS attackers:

Denial of QoS Service Request: The attacker can potentially intercept or drop all or some of the reservation messages such that the QoS reservation and channel setup can be failed or maliciously delayed in a persistent way.

Unnecessary/Suboptimal Resources Reservation: A particular user really wants to reserve X units of resources but the attacker causes the system to reserve Y units, while X is significantly different from Y.

Degradation of Network Utilization: The network system, as a whole, have enough resources to support a set of QoS requests, S_{req}. But, the attacker can, for instance, interfere with the reservation protocol such that the network can only support a small subset of S_{req}, $S_{partial}$.

Reserved QoS Degradation: Even if the resource reservation process is successful, the attacker can still steal the reserved resources. In other words, even the resources along the path has been reserved and maintained successfully, attackers can use the reserved resource unauthorizedly. Please note that, in most cases, the attacks that can achieve this objective are under the SECOND type of DoQoNS attacks, which is out of the scope of this paper.

The first three objectives can be achieved by attacking the control flow messages (*i.e.*, RSVP messages), while the last one is related to the refresh PATH/RESV messages, the dataflow packets, and the routing protocol itself.

3.2 Attackers

In network security, attackers are usually categorized into two different classes: *insider* and *outsider*. Outsider attacks can usually be prevented effectively by access control mechanism. However, insider attacks are much more difficult to deal with as an insider typically has some privileges to access certain trusted components.

For RSVP, we define three classes of attackers: $Insider_{RSVP}$, $Outsider_{OnPATH}$, and $Outsider_{Other}$. An $Insider_{RSVP}$ is a RSVP-enabled router on the reservation path between a sender and a receiver. Even if the network system is protected under a strong authentication and access control scheme, it is trusted to participate the RSVP message exchanges. $Outsider_{OnPATH}$, on the other hand, is a RSVP-disabled router also on the reservation path. Since it is not trusted to participate the RSVP operations, it can only intercept, delay, tamper (should be detected by some authentication mechanisms) or drop the RSVP messages. The weakest type of attackers is $Outsider_{Other}$. They are routers or end-hosts which are not on the reservation path. The rank of attacking power among these three classes is $Insider_{RSVP} \geq Outsider_{OnPATH} \geq Outsider_{Other}$ regardless of which protection schemes we choose to deploy. In other words, given a RSVP networking system with certain security countermeasures implemented and configured, any attack that can be performed by an $Outsider_{Other}$ can also be launched by either an $Insider_{RSVP}$ or an $Outsider_{OnPATH}$.

4 Attack Scenarios

4.1 Attacker's Targets and Operations

In RSVP, there are 7 objects or messages that can be the attacker's targets: `Tspec(`$PATH$`)`,`Adspec(`$PATH$`)`, `Rspec(`$RESV$`)`, `Tspec(`$RESV$`)`, `Filter_spec(`$RESV$`)`, `TearDown(`$PATH$`)`, `TearDown(`$RESV$`)`. These 7 types of objects or messages are included as part of either a PATH or RESV message. An attacker can perform any combination of the following operations on these objects: `Lower`, `Higher`, `Drop`, and `Inject`. For instance, an attacker can modify the value of `Adspec(`$PATH$`)` such that its value is "`Lower`" than it should be. Or, an attacker can "`Inject`" a `Tspec(`$RESV$`)` message to tear down successful reservation.

4.2 Four Scenarios, Eight Attack Examples

Some scenarios for RSVP control message attacks are presented in this section. These attacks can be performed by either an $Insider_{RSVP}$, $Outsider_{OnPATH}$, or $Outsider_{Other}$.

4.2.1 Scenario 1: `Tspec(`$PATH$`)` Tampering Attack

`Tspec(`$PATH$`)` defines the traffic characteristics of the dataflow that the sender will generate, and it is used by the traffic control module to prevent over-reservation. A vicious $Insider_{RSVP}$ router could silently modify the `Tspec(`$PATH$`)` in the incoming PATH messages. The receiver could make a wrong decision in reservation due to the faulty `Tspec(`$PATH$`)` information. If `Tspec(`$PATH$`)` is modified to a lower value (*i.e.*, the attacker performs a `Lower` operation on `Tspec(`$PATH$`)`), the quality of service that the receiver is going to enjoy might be degraded. On the other hand, if the attacker performs a `Higher` operation on `Tspec(`$PATH$`)`, then the receiver, depending on its local policy, may decide not to reserve the resources because the service looks more expensive (assuming some pricing mechanism is in place). In both cases, the receivers cannot enjoy the quality of service in a level that they should have while there are still plenty of resources available: Cases 1-1 and 1-2 in Figure 1.

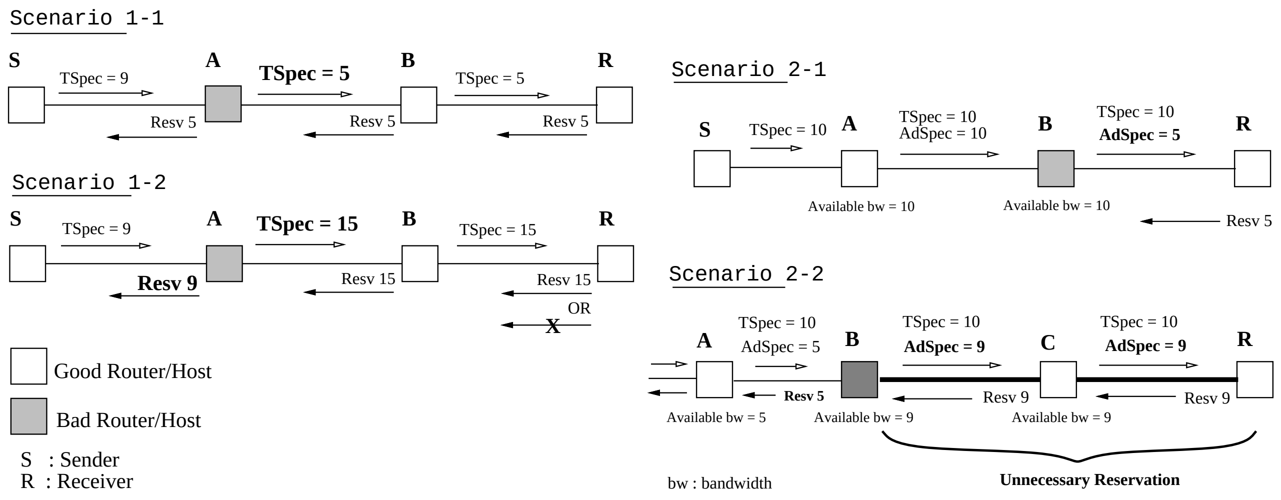

Figure 1: Insider Attack, Scenario 1 and 2

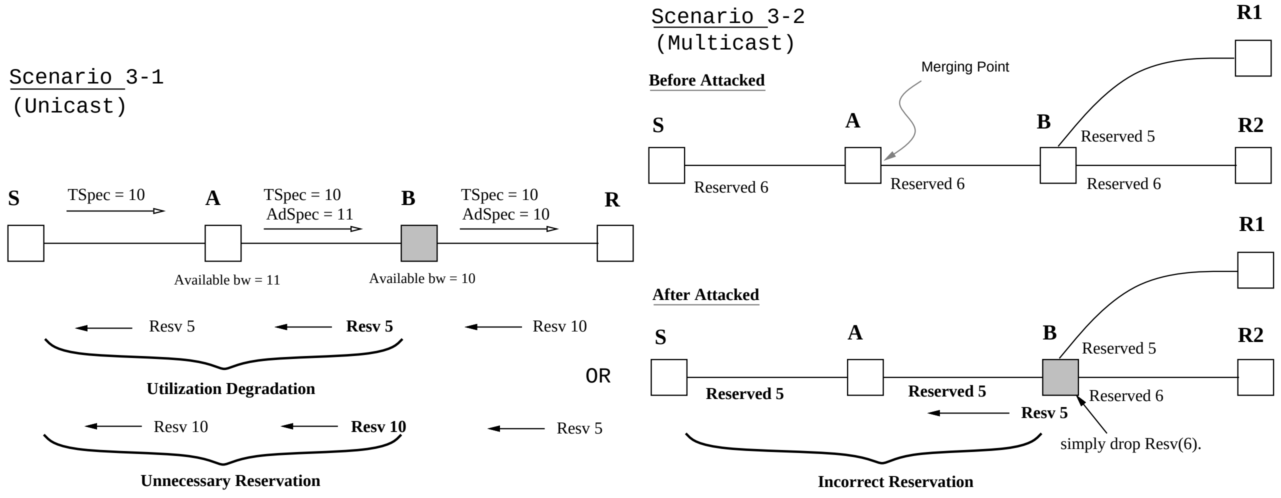

Figure 2: Insider Attack, Scenario 3-1 and 3-2

Furthermore, an aggressive attacker could modify both Tspec($PATH$) and Rspec($RESV$) objects to trick the receiver to make a much bigger (but unnecessary) reservation. Please note that an $Outsider_{OnPATH}$ (*e.g.*, node B in Scenario 1-3) can also perform the same attack causing same damage, but a $Outsider_{Other}$ can not because of the "soft state" refreshing mechanism in RSVP.

4.2.2 Scenario 2: Adspec($PATH$) Modification

The Adspec($PATH$) object is passed to the local traffic control module for Adspec($PATH$) update. Conceptually, the Adspec($PATH$) value represents the lowest resource availability among all the RSVP-enabled routers on the path. A vicious RSVP-enabled router can modify the value of Adspec($PATH$), which implies that all the RSVP-enabled routers downstream may update the Adspec($PATH$) incorrectly. Finally, the receiver may reserve a lower or higher level of QoS (*e.g.*, Scenario 2-1 in Figure 1).

4.2.3 Scenario 3: Rspec($RESV$) and Tspec($RESV$) Modification

As shown in Figure 2, parameters in the RESV messages (*e.g.*, Tspec($RESV$) or Rspec($RESV$)) [Wro97b] could be modified maliciously. These parameters are supposed to be untouched unless a merge is needed under the multicast case. The vicious RSVP-enabled router (or sometimes even $Outsider_{OnPATH}$) increases

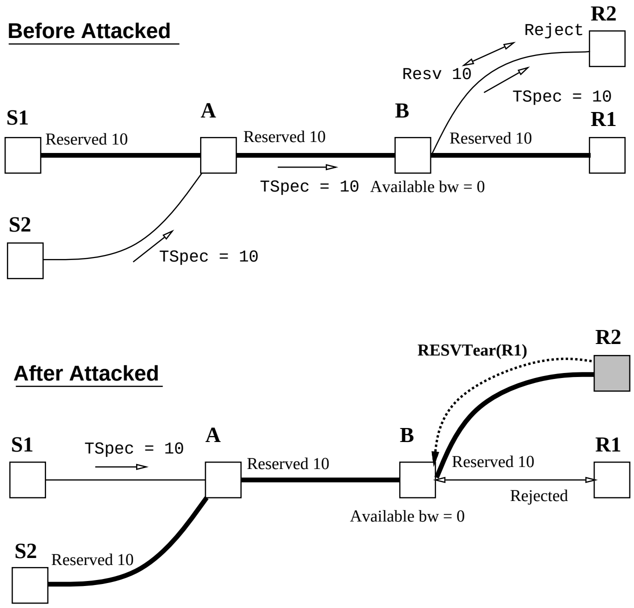

Figure 3: Insider Attack, Scenario 4

or decreases them unlegitimately such that the RSVP routers upstream might make unnecessary reservation or QoS degradation to receiver.

4.2.4 Scenario 4: Spoofed TearDown($RESV$)(Reservation TearDown)

TearDown($RESV$)messages are initiated explicitly by receivers or by any node in which reservation state has timed-out; they travel upstream towards all matching senders, but will cease to be forwarded at the node where merging would have suppressed forwarding of the corresponding RESV message. Receipt of a TearDown($RESV$)message deletes matching reservation state.

The TearDown($RESV$)message attack could be used by a vicious attacker when many users compete for limited resources on the network. In Figure 3, R1 and R2 belong to the same multicast group, and they share certain common security keys. Now, R2 tries to reserve resources for a session from *Sender S2* but it will fail because router B does not have enough resources to support R2's request. R2 generates an attack TearDown($RESV$)message to teardown R1's reservation. After R1's reservation is gone, R2 might have a very good chance to reserve the resources from router B. Please note that the victim receiver R1 might try to reserve the resources along the path again. But, this time it will fail because the resources have been committed to R2's session. Furthermore, both RSVP singalling operations and dataflow can be damaged simply by dropping attacks.

5 Defense

5.1 Difficulty

Although the soft-state mechanism supports some reliability for RSVP operation in a lossy, congested network environment. It's ability to against attacks, especially being launched by an $Insider_{RSVP}$ is limited. The integrity of RSVP messages is hard to protect due to the legitimate modifications to the message made by the RSVP-enabled routers on the path. While some objects such as Tspec($PATH$)are constants end-to-end, many others (Adspec($PATH$), PHOP, Rspec($RESV$), NHOP) are mutable legitimately. It is very difficult to tell whether a particular intermediate RSVP-enabled router is behaving correctly or not. A scheme similar to hop-by-hop authentication (such as [BLT98]) can not help because it assumes that all RSVP-enabled routers are trusted.

5.2 Selective Digital Signature and Conflict Detection (SDS/CD)

5.2.1 High-Level Description of SDS/CD

In order to deal with $Insider_{RSVP}$ attacks, we proposed to use a detection algorithm with modified end-to-end authentication. The basic idea is to separate the target RSVP objects into two different group: *constant* or *mutable*. For the constant part of the RSVP messages, the source or originator of the target object (*e.g.*, $Sender_{Alice}$ or $Receiver_{Bob}$) will digitally sign the object with its own private key. This prevents any $Insider_{RSVP}$ to tamper the signed object.

For the mutable part of the RSVP messages, we can not sign it because it will/might be changed, but, please note that they will NOT be changed AFTER the messages arrive the destination. For instance, a $PATH(sender \leftarrow receiver)$message might be changed before it arrives $Receiver_{Bob}$, but it should NOT be changed AFTER $Receiver_{Bob}$ gets it. The key point here is that mutable RSVP objects becomes "constant" after they become "history" of RSVP operations. Once they become "constant," $Receiver_{Bob}$ (for $PATH(sender \leftarrow receiver)$) or $Sender_{Alice}$ (for $RESV(receiver \leftarrow sender)$) can digitally sign the objects to commit the values they have just received.

Now, the "signed" history will be sent through the inversed route path, and every router on the path will have a chance (and optionally) to examine whether the history is consistent with their local observation. For instance, if $Router_{Chris}$ assigned a lower Adspec($PATH$)value than the "historical" version signed by $Receiver_{Bob}$, then we detect a conflict here: the final Adspec($PATH$)should be the lowest value along the route path. The response to this detected conflict depends on the local policy of the detecting RSVP-enabled router (*i.e.*, $Router_{Chris}$). For example, $Router_{Chris}$'s policy could specify that, if the difference between the local observation and the historical Adspec($PATH$)is greater than a threshold, then $Router_{Chris}$ will sign and issue a TearDown($RESV$)message toward $Receiver_{Bob}$. Since the TearDown($RESV$)message is digitally signed by $Router_{Chris}$, $Chris$ is hold accountable for this response.

An important question to answer is that whether the proposed approach can "detect" all types of tampering or malicious injecting. The answer is "NO," but a very interesting argument we have is that all the "SDS/CD-detectable" conflicts are much more critical than those "SDS/CD-undetectable" ones. Following our example in the previous paragraph, "increasing" viciously the value of Adspec($PATH$)by a RSVP-enabled router downstream is SDS/CD-detectable, but "decreasing" the Adspec($PATH$)is not. However, first, in RSVP, decreasing the Adspec($PATH$)value in the $PATH(sender \leftarrow receiver)$message is perfectly legal – this router conceptually does not have much resources to offer. Second, in fact, if a RSVP-enabled router is compromised or faulty, we should avoid using its services/resources anyway. "Decreasing" Adspec($PATH$)value will actually discourage sessions being established through this vicious or faulty router.

5.2.2 The SDS/CD Algorithm

The following is a more detailed description of the SDS/CD protocol:

1. $Sender_{Alice}$ selectively and digitally signs the Tspec($PATH$)(constant) part of the PATH message with its private key.

2. Upon receiving a PATH message, $Receiver_{Bob}$ verifies the integrity of Tspec($PATH$)the $Alice$'s public key, and then makes resources request accordingly.

3. $Receiver_{Bob}$ sends back RESV message **piggybacking** with, additionally, the Adspec($PATH$)object it just received (historical Adspec($PATH$)). Both the piggybacked Adspec($PATH$)and the Rspec($RESV$)in the RESV message are selectively and digitally signed with Bob's private key.

4. Upon receiving the RESV message, the intermediate RSVP-enabled router $Router_{Chris}$ verifies if the piggybacked (and signed) Adspec($PATH$)is equivalent to or lower than the last Adspec($PATH$)value it forwarded downstream. If it is not, a conflict has been detected and Chris will raise an alarm to its local policy decision point (PDP). The PDP will decide whether the RESV message should be dropped or not. Also, it will decide whether it is appropriate to flood a TearDown($RESV$)message downstream.

   In this case, the SDS/CD protocol at $Router_{Chris}$ detects that some vicious or faulty RSVP-enabled router downstream had reported an invalid Adspec($PATH$)value toward one of the receivers.

5. At an intermediate RSVP-enabled router, if a merge from multiple RESV messages is necessary, the router picks the RESV message with the largest Rspec($RESV$)value and forwards it toward $Sender_{Bob}$ (upstream).

6. When a RESV message arrives at $Sender_{Alice}$, the sender verifies if the Rspec($RESV$)is signed by one of the legitimate receivers. If it is not, the reservation should be cancelled. On the other hand, if it has been signed by a legitimate receiver, then that particular receiver is hold accountable for this reservation request.

7. Now, the Rspec($RESV$)value being received by the $Sender_{Alice}$ is "historical" (*i.e.*, no more merge should be performed). $Sender_{Alice}$ can digitally sign the Rspec($RESV$)value and piggybacks this signed and historical Rspec($RESV$)with the next refreshing PATH message.

8. Upon receiving the refresh PATH message, the intermediate RSVP-enabled router $Router_{Chris}$ checks if the piggybacked (and signed) Rspec($RESV$)is lower than the last Rspec($RESV$)value it forwarded upstream. If it is, a conflict has been detected and Chris will raise an alarm to its local policy decision point (PDP). The PDP will decide whether the PATH message should be dropped or not. Also, it will decide whether it is appropriate to flood a TearDown($PATH$)message upstream.

   In this case, the SDS/CD protocol at $Router_{Chris}$ detects that some vicious or faulty RSVP-enabled router upstream had chosen a smaller Rspec($RESV$)value in a merge operation and forwarded to the sender.

5.2.3 Properties about SDS/CD

For performance consideration, all detection or checking operations being performed in the intermediate RSVP-enabled routers can be optional. Also, some of the alarms can be forwarded to the sender(s) or receiver(s) (using DARPA's CIDF (Common Intrusion Detection Framework) or IETF's IDEX (Intrusion Detection Exchange Format), for example) such that they have the information and can decide whether to pick a new route or not.

In SDS/CD, only the sender or the receiver need to perform digital signature operations, while the other core routers only need to perform signature verification. Also, all such verification tasks are optional and can be performed in an off-line fashion. Furthermore, if there is no attacker, the information that needs to be signed may be very static (depending on also the network stability). Under such case, the sender or receiver can *pre-compute* the signature to speed up the performance of SDS/CD (*e.g.*, [MB96, HPT97]).

5.3 Related Proposed Protocols

Baker [BLT98] defines an INTEGRITY object to be carried in RSVP message in order to provide the information required for hop-by-hop integrity checking, *i.e.*, protecting RSVP message against spoofing and corruption. The contents of INTEGRITY object are Key Identifier, Sequence Number, and Keyed Message Digest. The Key Identifier is used to refer a secret Authentication Key and a keyed-hash algorithm. The monotonically increasing sequence numbers in INTEGRITY objects are used to prevent replay attack, a receiver only accepts RSVP messages which have a larger sequence number than the previous message. The digest of the whole RSVP message is written into the Keyed Message Digest field of the INTEGRITY object.

Some concerns in implementing this mechanism are as follows. First of all, the performance issue, due to the hop-by-hop mechanism, each RSVP router on the path would need to perform verification once and forward the original message if the authentication is successful. Under normal operation, the message issuer generates the INTEGRITY object, and the rest elements on the path verify it. However, since the Adspec($PATH$)objects in the $PATH(sender \leftarrow receiver)$messages are modified hop-by-hop mostly, the intermediate RSVP routers need to perform at least two message digest computations per RSVP messages. Furthermore, an attacker still can launch denial-of-service attacks by sending out fake RSVP messages with INTEGRITY objects. Exchanging RSVP Authentication Key regularly is suggested in order to maintain security. Furthermore, key exchange is also necessary when route changes occur because some "ex-participating RSVP routers" (insiders) might suddenly become outsiders and it is safe to use a brand new key. Unfortunately, it is not clear how the sender/receiver (for PATH/RESV messages) will know *when* and *where* a route change has occurred. In summary, hop-by-hop authentication is quite expensive and the key exchange process involved is also complicated and problematic.

Secondly, the hop-by-hop authentication is not useful in preventing insider attacks such as those we just mentioned in this paper. The effectiveness of hop-by-hop security checking is established on the trust relationship between two peer hops. If one rsvp-enabled router is compromised (or even just faulty), then the hop-by-hop scheme is not effective at all.

Finally, RSVP-enabled routers in Internet are likely to keep a common authentication key database (same copies, no matter how the mapping between Key Identifier and Authentication key is defined) in order to identify the RSVP messages with INTEGRITY objects came from any possible previous/next hop (PHOP/NHOP). The problem could be that if one of the RSVP routers is compromised, then the secret Authentication Key database is revealed and which makes the whole RSVP environment is vulnerable.

5.4 Evaluation

The following table shows the bad effects which can be caused by attacks. For instance, among the 8 attack examples, a simple scheme like reservation modification (i.e. attack 3-1 and 3-2) can cause three bad effects.

Attack Scenario	1-1	1-2	2-1	2-2	3-1 and 3-2	4	dropping
Unnecessary Reservation		V		V	V		
Network Utilization Degradation	V	V	V	V	V		
Reserved QoS Degradation					V	V	V
DoS to Receiver's Reservation						V	V

The next table summarizes that the effectiveness of *hop-by-hop authentication* versus *SDS/CD* in handling 8 attack cases:

Countermeasure	Hop-by-Hop	SDS/CD
Attack 1-1	$Outsider_{OnPATH}$	$Insider_{RSVP}$
Attack 1-2	$Outsider_{OnPATH}$	$Insider_{RSVP}$
Attack 2-1	$Outsider_{OnPATH}$	$Insider_{RSVP}$
Attack 2-2	$Outsider_{OnPATH}$	$Insider_{RSVP}$
Attack 3-1	$Outsider_{OnPATH}$	$Insider_{RSVP}$
Attack 3-2	$Outsider_{Other}$	$Insider_{RSVP}$
Attack 4	$Outsider_{OnPATH}$	$Insider_{RSVP}$
Dropping	$Outsider_{Other}$	$Outsider_{Other}$

Please note that, in Attack 3-2 (Figure 2), even with the hop-by-hop authentication, an $Outsider_{OnPATH}$ can drop the $RESV(6)$ message from $R2$ to router B to achieve the same result as B (a RSVP-enabled router) is compromised.

6 Remarks

We presented eight attack examples to demonstrate the vulnerabilities of the RSVP protocol in supporting network QoS. All these attacks can be performed by either an $Insider_{RSVP}$ or an $Outsider_{OnPATH}$ easily and yet they can cause significant damage on network QoS provision. The IETF/RSVP's security solution, hop-by-hop authentication, is only useful in preventing $Outsider_{OnPATH}$ (and in one case, only preventing $Outsider_{Other}$, which is the weakest attacker model). It can not handle any $Insider_{RSVP}$ attacks.

The SDS/CD (Selective Digital Signature with Conflict Detection) protocol we proposed addresses a fundamental issue in network security: *how can we protect, in an End-to-End fashion, the integrity of an object that is mutatable along the route path?* Because of this difficult problem, hop-by-hop authentication scheme (which compromised the security of RSVP) has been adopted within IETF/RSVP. SDS/CD resolved this fundamental issue by delaying the integrity protection process until the target object becomes *history* and letting the intermediate RSVP-enabled routers, which could have modified the target object's value, to detect the conflicts. We showed that, under SDS/CD, many types of $Insider_{RSVP}$ attacks can be handled in a flexible, secure and efficient way.

SDS/CD still can not handle the RSVP message dropping attacks, which probably are not *preventable*. We have developed different intrusion detection techniques (*e.g.*, [CL97]) in locating the dropping point. We believe that the combination of various prevention and detection schemes together will hopefully provide a strong solution in protecting network QoS.

Acknowledgments

Liang Li and Steve Nadas from IBM/RTP have generously provided us a beta implementation of RSVP running on IBM's 2210 routers, which we used to perform some of our experiments.

References

[BBB+98] Y. Bernet, J. Binder, S. Blake, M. Carlson, E. Davies, B. Ohlman, D. Verma, Z. Wang, and W. Weiss. A Framework for Differentiated Services. Internet Draft, IETF, draft-ietf-diffserv-framework-00.txt, May 1998. Network Working Group.

[BBC+98] D. Black, S. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. Internet Draft, IETF, draft-ietf-diffserv-arch-00.txt, May 1998. Network Working Group.

[BLT98] F. Baker, B. Lindall, and M. Talwar. RSVP Cryptographic Authentication. Internet Draft, IETF, November 1998. Network Working Group.

[CL97] S. Cheung and K.N. Levitt. Protecting Routing Infrastructure from Denial of Service Using Cooperative Intrusion Detection. In *New Security Paradigms Workshop*, Cumbria, UK, September 1997.

[HPT97] R. Hauser, T. Przygienda, and G. Tsudik. Reducing the Cost of Security in Link-State Routing. In *Internet Society Symposium on Network and Distributed Systems Security*, 1997.

[MB96] S.L. Murphy and M.R. Badger. Digital Signature Protection of the OSPF Routing Protocol. In *Internet Society Symposium on Network and Distributed Systems Security*, 1996.

[SPG97] S. Shenker, C. Partridge, and R. Guerin. Specification of the Guaranteed Quality of Service. RFC 2210, IETF, September 1997. Network Working Group.

[Tal98] M. Talwar. RSVP Killer Reservations. Internet Draft, IETF, November 1998. Network Working Group.

[Wro97a] J. Wroclawski. Specification of the Controlled Load Quality of Service. RFC 2211, IETF, September 1997. Network Working Group.

[Wro97b] J. Wroclawski. The Use of RSVP with Intergrated Services. RFC 2210, IETF, September 1997. Network Working Group.

[YPG98] R. Yavatkar, D. Pendarakis, and R. Guerin. A Framework for Policy-based Admission Control. Internet Draft, IETF, November 1998. Network Working Group.

[ZBE+97] L. Zhang, B. Braden, D. Estrin, S. Herzog, and S. Jamin. Resource ReSerVationProtocol (RSVP) Version 1 Functional Specification. RFC 2205, IETF, September 1997. Network Working Group.