Optimum Security Service for Heterogeneous Multicast Receivers

Antony Sagaya Jeyanthi, K.C.Nishitha

Abstract*— The Resource Reservation Protocol (RSVP) lets hosts request quality of (bandwidth) service for multicast applications on the Internet. As network equipment advances to provide improved bandwidth service, security service becomes the more critical problem. However, RSVP doesn't provide a flexible mechanism to support quality of security service (QoSS). Security service RSVP extends RSVP to provide the needed mechanism for dynamically negotiating QoSS among the senders and heterogeneous receivers of multicast applications on the Internet with minimum overhead. SSRSVP provides different QoSS resolutions according to receiver nodes' security service needs.

Index Terms*— Multicast, Quality of Security Service, Resource ReSerVation Protocol.

I. INTRODUCTION

The Resource Reservation Protocol (RSVP) [2, 3] lets hosts request specific quality of service (QoS) from the network for multicast multimedia flows on the Internet.[2,4,5] Both multiprotocol label switching and generalized MPLS architectures [6,7] use RSVP to establish label switched paths.[8,9] Dynamic RSVP (DRSVP) provides a more flexible and dynamic resource-reservation mechanism, [10] and other works have successfully solved resource reservation and negotiation in virtual private networks.[11–13]

However, although RSVP provides quality of bandwidth service, it doesn't support *quality of security service* (QoSS).[14] Cynthia Irvine and Timothy Levin, the term's originators, focused on QoSS from system administrator and user viewpoints. We refine QoSS to include the security-service negotiations among senders and receivers in a network — that is, security-service multidimension spaces composed of strength of cryptographic algorithms, length of cryptographic key, robustness of authentication mechanisms, and so on, negotiated among senders and receivers on the Internet.

Security-service negotiations between sender and receiver often use session mechanisms or protocols.[15] Traditional session mechanisms and protocols between sender and receiver are suitable for point-to-point communication, in which senders transmit negotiation requests to receivers. However, session mechanisms and protocols can't solve the QoSS problem when users have different security service needs and security-processing capabilities. For example, if the sender can only cope with the data encryption standard (DES) algorithm, and the receiver can only cope with the RSA algorithm, they must depend on a third party to implement a conversion function between them.

This problem is multiplied for multicast applications on the Internet. Multicast enables efficient large-scale content distribution by providing an efficient transport mechanism for one-to-many and many-to-many communication. Although many security mechanisms enhance multicast application security, these tools generally assume that receivers and senders use the same security service (that is, the same algorithms). If, for example, one sender communicates with *N* different receivers that have different security-processing capabilities (that is, security algorithms), the sender must transmit the *N* different packets using *N* different security services. When *N* is large, it's difficult for the sender to communicate with receivers. In the worst-case scenario, the sender doesn't have any algorithms matching those of the receivers, so it can't communicate with them using general security mechanisms. Thus, QoSS for multicast Internet applications is an important research area.

Security service RSVP dynamically negotiates QoSS among the senders and receivers of multicast Internet applications. [1] SSRSVP differs from the mechanisms described in RFC 2747[16] and RFC 2207, [13] which use RSVP's integrity object to provide hop-by-hop integrity and authentication of RSVP messages and support IP security (IPsec).[11–13] Our extension doesn't aim to enhance or support RSVP's security function, but to solve QoSS negotiation among senders and receivers of multicast applications with different security service needs and security-processing capabilities. I don't discuss key transmission, error message, tear message, and soft state in the article due to space limitations.

II. SSRSVP MODEL

Figure 1 shows the SSRSVP model, which is similar to RSVP1 and DRSVP9 models. Admission control determines whether the node has sufficient security-processing capability to support the negotiation request, and policy control determines whether the user has sufficient privilege to make the request. If the answer to both control modules is passed, the corresponding SSRSVP parameters are set.

SSRSVP defines a session as a packet flow and treats each session independently. SSRSVP sessions include an IP destination address, protocol ID, and destination port. The classifier module determines the QoSS level for each packet in the host or router. The security process module implements each packet's security processing, including confidentiality and integrity functions.

Manuscript received on July, 2013.

Ms.S.Antony Sagaya Jeyanthi, Information Technology department,Veltech Multitech Dr.Rangarajan Dr.Sakunthala Engineering College,Chennai,India.

Ms.K.C.Nishitha, Information Technology department,Veltech Multitech Dr.Rangarajan Dr.Sakunthala Engineering College,Chennai,India.

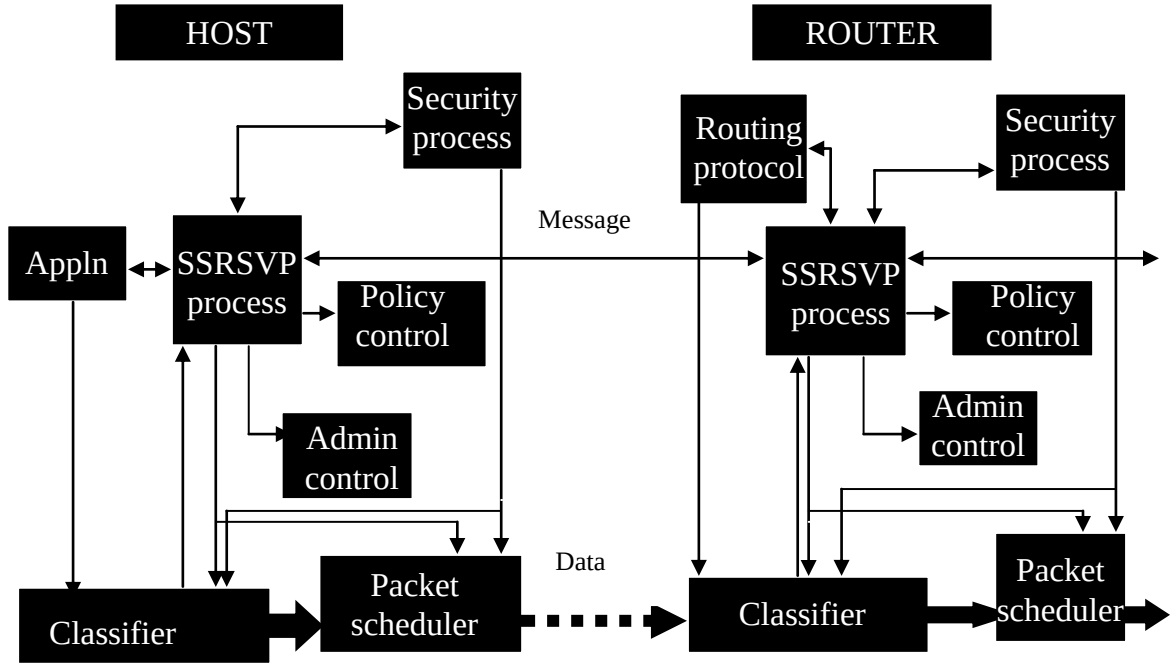

Figure 1. **The security service Resource Reservation Protocol (SSRSVP) model in hosts and routers**. The security process module implements each packet's security processing based on information it extracts from the SSRSVP quality of security service (QoSS) database .

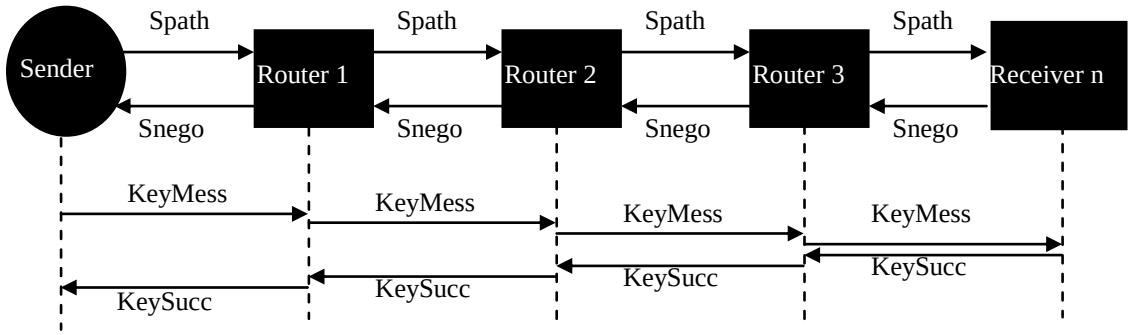

Figure 2. **Security service mechanism**. Key messages containing negotiation acknowledgment and Quality of security service (QoSS) information travel from the sender, through several routers, to the receiver. Upon receipt of the KeyMess, the receiver returns a key success (KeySucc) message and begins to transmit the requested data along the same path

When a packet arrives at a router without an SSRSVP module, the router checks the packet's route table information and forwards it.

When a packet arrives at a router that has an SSRSVP module and the SSRSVP setup is complete, the router implements several operations. First, the classifier module determines the packet's QoSS level according to its IP destination address, protocol ID, and destination port. Next, the security process module extracts information from the SSRSVP QoSS database according to the packet's QoSS. This information includes keys, cryptography algorithms, and QoSS information negotiated with the next hop. The security process module then reencapsulates the packet according to the QoSS information negotiated with the next hop, and the router forwards the packet to the next hop according to the route table information.

Table 1. Security service RSVP message types.

1 **Spath** SSRSVP path
2 **Snego** SSRSVP negotiation request
3 **SpathErr** SSRSVP path error
4 **SnegoErr** SSRSVP negotiation error
5 **SPathTear** SSRSVP path tear
6 **SnegoTear** SSRSVP negotiation tear
7 **SnegoConf** SSRSVP negotiation confirmation
8 **KeyMess** Key message
9 **KeyFail** Key message failure
10 **KeySucc** Key message success

III. SECURITY SERVICE MECHANISM

Like RSVP1 and DRSVP, [9] SSRSVP adopts a receiver initiated design principle: Receivers choose the QoSS negotiation request and initiate the negotiation, keeping it

active for as long as they want to receive the packet.Figure 2 shows SSRSVP's three fundamental
message types: SSRSVP negotiation (**Snego**), SSRSVP path (**Spath**), and key message (**KeyMess**).

SSRSVP senders transmit **Spath** messages downstream along the unicast or multicast routes provided by the routing protocols, following the packets' paths. These **Spath** messages store the path state in each node along the way. The path state includes the unicast IP address of the previous hop node, which is used to route the **Snego** messages hop-by-hop in the reverse direction.

SSRSVP receivers send **Snego** messages upstream to all senders in the sender selection. These messages must follow exactly the reverse of the path the packets will use.

When the sender finishes the SSRSVP negotiation request, it transmits a **KeyMess** downstream along the path to the next hop. Figure 2 illustrates this process:

1. The sender transmits the **KeyMess** to *router*1.The message includes negotiation acknowledgment information and information about the QoSS that *end router(router3)* will use to decrypt data packets encrypted by the sender. If the **KeyMess** transmission fails, a key failure message(**KeyFail**) is returned to the sender; otherwise, the operation continues.
2. *Router*1 forwards the **KeyMess** sent by the sender to the *router2*. If the process fails, *router*2 returns the **KeyFail** to *router*1, and *router*1 returns it to the sender. If the process succeeds, *router*2 repeats this step with *router*3.
3. *Router*3 attempts to transfer the **KeyMess** to the receiver. The message includes negotiation acknowledgment information and information about the QoSS that *receiver* will use to decrypt data packets encrypted by the end router(router3). If router3 successfully transfers the **KeyMess** to the receiver, the receiver returns a key success message to *router*3, which then returns **KeySucc** to *router*2, and so on. When the sender receives the **KeySucc**, it begins to transfer the data along the path to the receiver.

At each intermediate router, an **Spath** message triggers three general actions (when an **Spath** message passes a router's authentication and policy control, it can also trigger these actions).

First, the intermediate router adds the previous hop address in the **Spath** message's **SSRSVP_Hop** field. The intermediate router stores the information extracted from the **Spath** message. This information includes the sender template, which describes the format of the packet that the sender will originate. The sender template specifies the sender IP destination address, protocol ID, and destination port. The sender's security properties specify its maximal and minimal security-processing capability and the QoSS-level limit for the packet's intended receivers.

Next, the intermediate router forwards the **Spath** message downstream toward the receivers along the unicast or multicast routes provided by the routing protocols. At each border router connected with a receiver and an intermediate router, a **Snego** message triggers the following two general actions, respectively.

First, the receiver passes the **Snego** request to the admission- and policy-control engines of the router SSRSVP. If either test fails, the router rejects the negotiation request and the SSRSVP process returns an SSRSVP negotiation error (**SnegoErr**) message to the receivers. If both tests succeed, the router returns a confirmation message to the receivers. This message indicates the network's QoSS. If the sender's **Spath** message claims a QoSS-level limit, the router checks the limit against the QoSS level in the receiver's negotiation request. If the negotiation request level is under the sender's limit, the router rejects the negotiation request. When the receiver gets a **SnegoErr** message from the router, it sends a negotiation request with a higher QoSS level if it has a higher security-processing capability. If not, it can't receive packets from the sender. Next, the intermediate router forwards the **Snego** message upstream. It makes QoSS negotiation emergence according to the receiver's negotiation style and then propagates a new negotiation request message upstream toward the sender.

IV. NEGOTIATION STYLES

From an RSVP viewpoint, senders can transmit data whether or not the network provides adequate bandwidth service to deliver the data. However, QoSS differs from quality of bandwidth service in that senders don't want to transmit messages to receivers that don't have adequate security because this might expose their messages to others or let others forge them during transmission. A sender could choose to transfer packets to receivers with QoSS levels that are above its QoSS limit, thereby forcing receivers to keep high negotiation-request QoSS levels. Of course, the sender will always use the maximum security- processing level to encapsulate data packets and forward them to the next hop router with SSRSVP regardless of that hop's negotiation request level. We use negotiation styles to guarantee receivers' QoSS levels.

We classify SSRSVP negotiation styles into two categories:

- *limit* (limit wild card-filter [LWF], limit fixed filter [LFF], and limit shared-explicit [LSE] negotiation styles) and
- *non limit* (wild card-filter [WF], fixed-filter [FF], and shared-explicit [SE] negotiation styles).

Let a negotiation request be SQ_i, where SQ_i denotes the QoSS level of the i^{th} receiver's negotiation request.

$Infs_i$ ( ) means that the QoSS level negotiated by the receivers isn't under the *i*th sender's limit. Because SSRSVP's non limit negotiation styles (WF, FF, and SE) are the same as RSVP's reservation style, [2, 10] we don't describe them further. We define the negotiation style as follows:

• LWF style: (* $Max[Infs_i (SQ)]$) and
• WF style: (* $Max(SQ))_{i\_} (1 \ldots n)$.

During the LWF-style negotiation process, the router merges separate negotiation requests into one negotiation request for each upstream sender:

- LFF style: ($S1$ $Max[Infs_i (SQ)]$), $S2$ $Max[Infs_2 (SQ)]$), and
- FF style: ($S1$ $Max(SQ_i)$, $S2$ $Max(SQ_j)$) $(i, j\_ (1 \ldots n))$.

Assume that $S1$ and $S2$ denote senders 1 and 2.The LFF-style negotiation request creates a distinct negotiation for data packets from a particular sender, not sharing them with packets from other senders in the same session:

- LSE style :(($S1$, $S2$) {$Max\ Infs_{i,}(Infs2, (SQ_{i)})$]}) and

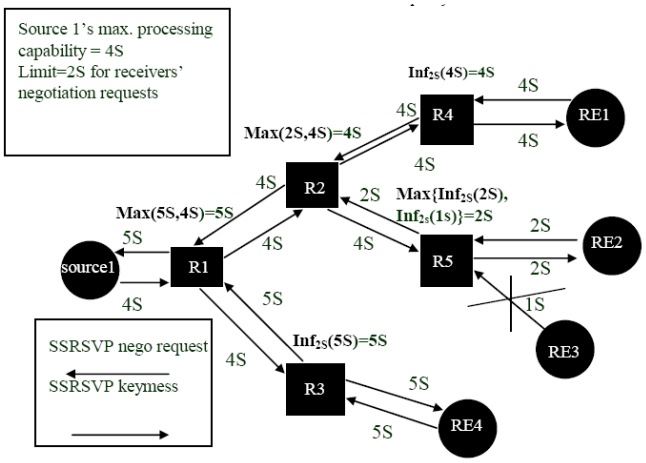

Figure 3. A complete security service Resource Reservation Protocol (SSRSVP) negotiation request and **KeyMess**. The request comes from a single source node to four receiver nodes using limit fixed-filter (LFF) style.

V.SSRSVP EXAMPLE

We define the QoSS levels used in the SSRSVP examples in Figures 3 and 4 as follows:

• 1*S* is first-level QoSS (using, for example, DES and Message-Digest algorithm 5, or MD5);
• 2*S* is second-level QoSS (3DES and MD5);
• 3*S* is third-level QoSS (1,024-bit RSA and MD5);
• 4*S* is fourth-level QoSS (1,024-bit RSA and Secure Hash Algorithm, or SHA); and
• 5*S* is fifth-level QoSS (2,048-bit RSA and SHA).

Figure 3 shows a single source node sending a negotiation request to four receiver nodes using the LFF (Figure 3) styles.

Because SSRSVP is a receiver-oriented request protocol, receivers *R*1, *R*2, *R*3, and *R*4 will initiate an LFF-style SSRSVP request. Suppose *router*1, *router*2, *router*3, *router*4, and *router*5 have available security-processing capability to support the SSRSVP negotiation request. Assume the QoSS levels of negotiation requests from *R*1, *R*2, *R*3, and *R*4 are 4*S*, 2*S*, 1*S*, and 5*S*, respectively.

In Figure 3, because *R*3's negotiation-request level is under *S*1's limit, *router*5 rejects *R*3's negotiation request.

- SE style:((*S*1,*S*2){*Max*(*SQi*)})(*i* _(1 … *n*)).

The LSE-style negotiation request creates a single negotiation shared by selected upstream senders. Unlike the LWF style, the LSE style lets receivers explicitly specify the set of senders to include.

If *R*3's maximum QoSS level is 1*S*, it will never receive data packets from *S*1. Of course, if 1*S* isn't *R*3's maximum security-processing capability, it will try to send a negotiation request with high-level QoSS to *router*5 when it receives the **SnegoErr** message.

When the negotiation request is finished, the sender node *S*1 sends a **KeyMess** to router1 which includes negotiation acknowledgment information and the QoSS information that end router (the router which is connected with the receiver) uses to decrypt data packets encrypted by the sender. When *router*1 receives the **KeyMess** from *S*1, it will be forwarded to the end router through the intermediate routers. When it reaches end router it sends the **KeyMess** to the receiver uses to decrypt data packets encrypted by the end router. When all of the receiver nodes receive the upstream **KeyMess** message, SSRSVP setup is successful. It will reduce the additional work of decryption and encryption that has been done in intermediate routers while transmitting multicast application to the receivers ,which are having different security service needs and capabilities.

VI. APPLICATION SCENERIO

For multicast applications on the Internet, users in a multicast group can use different security services. Users in a malicious network environment domain will
want data sent to them using high level QoSS (that is, more complex security algorithms to encrypt, authenticate, and protect integrity). Users in a secure network environment domain, however, will want to improve their data processing performance by using low- or mid-level QoSS. SSRSVP guarantees simultaneity while providing for these users' different needs.

For example, in a military information war scenario, military officers of varying ranks and general soldiers in a multicast group might have QoSS levels assigned according to their ranks. General soldiers can only process data with first-level QoSS; the company commander can process data with first- or second-level QoSS; the colonel can process data with first-, second-, or third-level QoSS; and the brigadier can process data with all QoSS levels. Soldiers and military officers can communicate with each other. Sometimes, however, the colonel will want to communicate command information only to officers whose military rank isn't under company-commander level. SSRSVP makes this possible.

VII. CONCLUSION

Security Service Resource ReSerVation Protocol (SSRSVP) solves the security problem arise when users have different security service needs and security processing capabilities in multicast application on Internet .It provides a flexible mechanism to support quality of security service (QoSS) using RSVP in multicast applications. And by eliminating additional work regarding security in sender and intermediate routers, it minimizes the overhead in transmitting secure multicasting application**.** This will include defining Quality of Security Service (QoSS) architecture for Internet networks, providing end-to-end security-service delivery and hop-to-hop security-service safeguards, and formalizing QoSS.

REFERENCES

1. ZhengYou Xia, YunAn Hu , "Extending RSVP for Quality of Security Service", "IEEE Internet Computing, vol.10, no. 2,March/April 2006,pp.51-57.
2. L. Zhang et al., "RSVP: A New Resource Reservation Protocol," IEEE Network, vol. 7, no. 5, Sept. 1993, pp. 8–18.
3. R. Braden et al., Resource Reservation Protocol (RSVP) —Version 1 Functional Specification, IETF RFC 2205, Sept.1997, www.ietf.org/rfc/rfc2205.txt.
4. R. Braden, D. Clark, and S. Shenker, Integrated Services in the Internet Architecture: An Overview, IETF RFC 1633,June1994; ww.ietf.org/rfc/rfc1633.txt.
5. J. Wroclaw ski, The Use of RSVP with IETF Integrated Services,IETF RFC 2210, Sept.1997, www.ietf.org/rfc/rfc2210.txt.
6. E.C. Rosen, A. Viswanathan, and R. Callon, Multiprotocol Label Switching Architecture, IETF RFC 3031, Jan. 2001,www.ietf.org/rfc/rfc3031.txt.
7. E. Mannie, ed., "Generalized Multiprotocol Label Switching (GMPLS) Architecture," IETF Internet draft, work in progress, Aug. 2002.
8. D. Awduche et al., RSVP-TE: Extensions to RSVP for LSP Tunnels, IETF RFC 3209, Dec. 2001, www.ietf.org/rfc/rfc3209.txt.

9. L. Berger, ed., Generalized Multiprotocol Label Switching (GMPLS) Signaling Resource Reservation Protocol-Traffic Engineering (RSVP-TE) Extensions, IETF RFC 3473, Jan. 2003; www.ietf.org/rfc/rfc3473.txt.
10. G.-S. Kuo and Po-Chang Ko, "Dynamic RSVP Protocol," IEEE Comm., vol. 41, May 2003, pp. 130–135.
11. F. Baker and P. Bose, "QoS Signaling in a Nested Virtual Private Network," IETF Internet draft, work in progress, Oct.2005.
12. B. Pratik, D. Voce, and D. Gokhale, "QoS for Aggregated Flows in VPN," Proc. Int'l Workshop Quality of Service(IWQOS), LNCS 3552, Springer-Verlag, 2005, pp. 392–394.
13. L. Berger and T. O'Malley, RSVP Extensions for IPSec Data Flows, IETF RFC 2207, Sept. 1997; www.ietf.org/rfc/rfc2207.txt.
14. C. Irvine and T. Levin, "Quality of Security Service," Proc.New Security Paradigms Workshop, ACM Press, 2000, pp.91–99.
15. M. Handley and V. Jacobson, SDP: Session Description Protocol, IETF RFC 2327, Apr. 1998; www.ietf.org/rfc/rfc2327.txt.
16. F. Baker, B. Lindell, and M. Talwar, RSVP Cryptographic Authentication, IETF RFC 2747, Jan. 2000.

Ms. S. Antony Sagaya Jeyanthi received the B. Tech degree in Information Technology from Francis Xavier Engineering College, Tirunelveli ,Tamil Nadu,India and the M.E degree in Computer Science and Engineering from PSNA College of Engineering and Technology, Dindigul, Tamil Nadu,India and started to do PhD under Veltech University ,Chennai,Tamil Nadu, India, in 2006,2008,2013 receptively.

She is currently an Assistant Professor with the Veltech Multitech Dr.Rangarajan Dr.Sakunthala Engineering College,Avadi,Chennai,Tamil Nadu,India. Her current research interest includes Network Security and Routing Algorithm and Wireless Network.

Ms. K.C.Nishitha received the B.E degree in Computer Science and Engineering from Vins Christian College of Engineering, Nagercoil ,Tamil Nadu,India and the M.E degree in Computer Science and Engineering from Anand Institute of Higher Technology and Technology, Chennai, Tamil Nadu,India in 2010,2012 receptively.

She is currently an Assistant Professor with the Veltech Multitech Dr.Rangarajan Dr.Sakunthala Engineering College,Avadi,Chennai,Tamil Nadu,India. Her current research interest includes Networks and Data Mining.