Deploying RSVP in Multiple Security Domains Networks: Securing Application Quality of Service

Executive Summary

This white paper brings out a possible solution for deploying Resource Reservation Protocol (RSVP) admission control within a multiple security domains network.

Multiple security domains networks, as they are deployed today, often lack strict QoS guarantees, deterministic performance, application-aware and user-aware policy enforcement, and efficient bandwidth utilization. RSVP-based admission control solution is a widely accepted mechanism to provide all these benefits in an enterprise environment. However, deploying RSVP inside a multiple security domains network requires special attention to achieve end-to-end resource reservation across the security boundaries.

This white paper is organized in the following manner. We start by providing an overview of the RSVP protocol; next we attempt to illustrate the issues of deploying RSVP in multiple security domains networks. Subsequently, we discuss the different building blocks that can be part of the overall solution. Finally we describe an architecture that combines these building blocks and allows end-to-end RSVP-based admission control and reservation in a multiple security domains network.

RSVP Overview

RSVP is a network control protocol used to provide guaranteed quality of service for specified application data flows. It is used for real-time applications such as voice and video that need predictable service requirements from the network (that is, minimized delay, jitter, and loss). The base RSVP protocol is documented in RFC 2205.

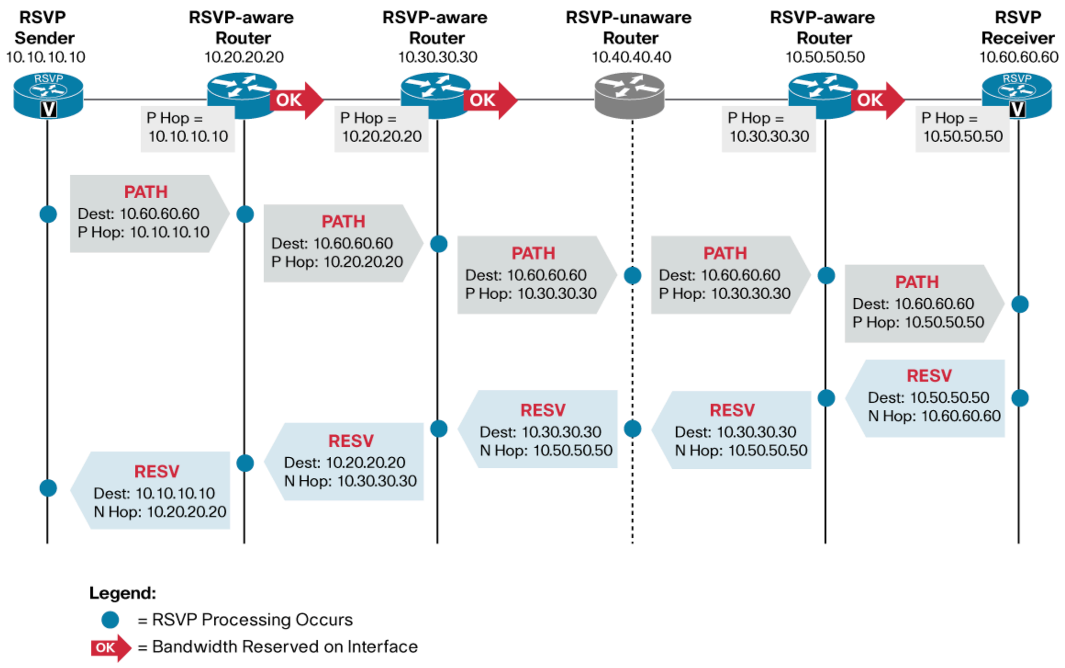
RSVP is a signaling mechanism that is initiated by hosts to request certain bandwidth and performance guarantees from the network on behalf of a particular application running on that host. The sender host that will behave as the source for the application flow (sender) starts the RSVP process by sending an RSVP Path message towards the host that will behave as the receiver for the flow (receiver). The Path message is processed on every RSVP router. The RSVP router establishes a state storing the information contained in the Path message (Path state). Then the RSVP router forwards the Path message using whatever existing routing protocols are already running on the router to determine the data path for the actual flow that is protected by RSVP. Before transmitting the Path message, the RSVP router includes its IP address in the Path message (in the RSVP Previous Hop field). This effectively tells the next RSVP router who sent that message.

On receipt of the Path message, the receiver host will issue an RSVP Resv message back towards the sender. The Resv message is the one triggering the actual reservation of resources in the network. The Resv message is processed by every RSVP router. First the RSVP router performs two local decision mechanisms: bandwidth admission control and policy control. For admission control, parameters such as bandwidth are checked to determine whether the node has the required bandwidth specified in the Resv message before the RSVP request is admitted. For policy control, additional rules are checked by the RSVP router before admitting the reservation. This allows enforcement of user-aware or application-aware policy rules (for example, video cannot use more than 60 percent of link bandwidth, a particular user has higher priority for reserving resources). The policy control rules can be configured locally on the router or on a centrally located policy server.

If either the admission control or the policy control fails for the Resv message, a ResvErr (Resv Error) message is sent back to the receiver that requested reservation, and the reservation request fails. Assuming that both the admission control and the policy control are successful for the Resv message, then a state storing the information contained in the Resv message is created (Resv state) by the RSVP router. The RSVP router sets aside the bandwidth allocated for that reservation (and, if needed, dynamically programs the requested quality of service [QoS] in the packet scheduler and packet classifier). The Resv message is forwarded to the next RSVP router towards the sender by using the RSVP Previous Hop information stored locally in the Path state.

This sequence for the Resv message is repeated on each RSVP router until it reaches the sender. This means that admission was successful at each RSVP router along the path, and therefore there is an end-to-end reservation in place from sender to receiver for the flow of interest. Please see Figure 1 for better illustration. From then on, each packet that meets the packet classification parameters specified in RSVP inside a flow specification for the reservation is sent using the reserved QoS resources. All other packets are sent using best-effort delivery.

Figure 1. End-to-End RSVP Reservation


RSVP Sender 10.10.10.10
RSVP-aware Router 10.20.20.20
RSVP-aware Router 10.30.30.30
RSVP-unaware Router 10.40.40.40
RSVP-aware Router 10.50.50.50
RSVP Receiver 10.60.60.60
OK
P Hop = 10.10.10.10
P Hop = 10.20.20.20
P Hop = 10.30.30.30
P Hop = 10.50.50.50
PATH Dest: 10.60.60.60 P Hop: 10.10.10.10
PATH Dest: 10.60.60.60 P Hop: 10.20.20.20
PATH Dest: 10.60.60.60 P Hop: 10.30.30.30
PATH Dest: 10.60.60.60 P Hop: 10.30.30.30
PATH Dest: 10.60.60.60 P Hop: 10.50.50.50
RESV Dest: 10.50.50.50 N Hop: 10.60.60.60
RESV Dest: 10.30.30.30 N Hop: 10.50.50.50
RESV Dest: 10.30.30.30 N Hop: 10.50.50.50
RESV Dest: 10.20.20.20 N Hop: 10.30.30.30
RESV Dest: 10.10.10.10 N Hop: 10.20.20.20
Legend:
● = RSVP Processing Occurs
OK = Bandwidth Reserved on Interface

As described above, RSVP is unidirectional, that is, it establishes reservation for one direction. If the application requires reservation in both directions, then the RSVP process described above would happen independently for each direction.

The RSVP control protocol supports both unicast and multicast reservations. For multicast, the protocol scales well in large multicast groups since it uses receiver-oriented requests that merge together as you move up the multicast tree towards the sender. Much of the complexity of the protocol is due to the requirement that multicast must be supported.

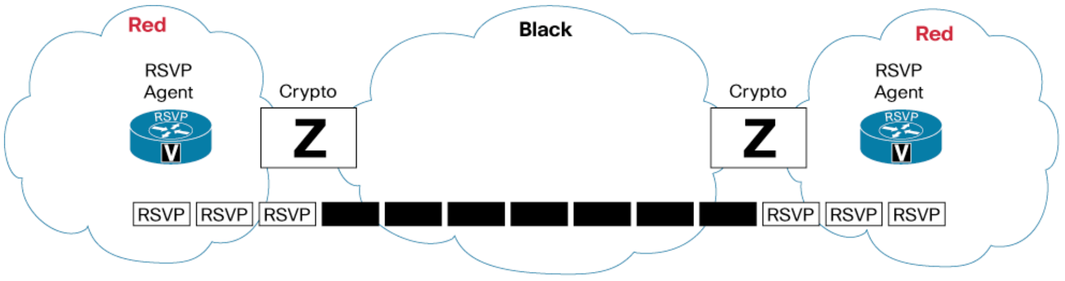
One important property of the RSVP protocol is that it will be forwarded transparently by a router that does not support RSVP (or a router on which RSVP has not been enabled). This was a design choice to allow incremental deployment of RSVP. Thus, RSVP can be activated selectively only on routers where its benefits (of admission control and resource reservation) are needed. This is illustrated in Figure 1, which includes an RSVP-unaware router.

Many extensions have been defined since the initial RSVP specification. Those efforts have considerably broadened the applicability, scalability, and security of RSVP. For example, this includes extensions to RSVP (for example, RFC 3209) for support of MPLS traffic engineering (RSVP-TE) technology that is very widely deployed in many large-scale service provider networks worldwide. This also includes operations of RSVP over Diffserv clouds (for example, RFC 2998), which allows RSVP reservation to take advantage of Diffserv mechanisms in the datapath, thereby considerably improving RSVP scalability.

RSVP Deployment Issue with Multiple Security Domains Network

Figure 2 illustrates the basic issue of deploying RSVP in a multiple security domains network if no special care is taken for RSVP. As a quick overview, when information is handled within a security domain (enclave), it is often referred to as being "in the clear." This simply means that the information is neither encrypted nor protected in any fashion other than the physical security inherent to security enclave separation. Networks that carry information in the clear are designated as red networks, while networks that carry only encrypted information are designated as black networks. The RSVP messages originate in the red side of the network and proceed through a crypto gateway that encrypts all packets and sends them into a VPN tunnel; we effectively have RSVP messages being hidden while transiting the black cloud, and then they reappear on the other end of the VPN tunnel. In our example, only the RSVP agents on both ends (and RSVP routers along the reservation path inside the red network) are processing the RSVP messages; therefore, we essentially can't have hop-by-hop bandwidth reservation in the black network. This defeats the purpose of deploying RSVP to secure end-to-end bandwidth reservation.

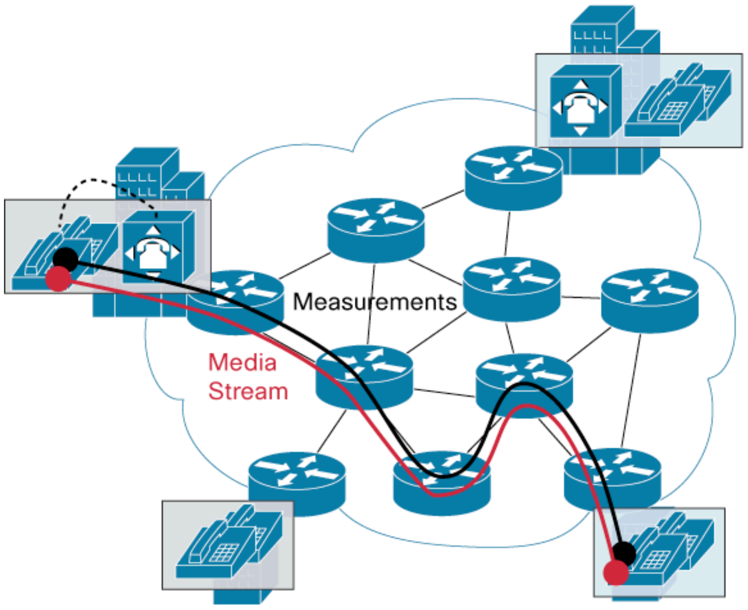
Figure 2. RSVP Deployment Issue in Multiple Security Domains Network



Measurement-based Alternative Solutions without RSVP

There are solutions being deployed or considered today to mitigate this issue, as depicted in Figure 3. Most of these solutions rely on end or edge devices to observe and measure performance of the network. As an example, before a voice call is made, measurement is made to make sure that the delay is within the tolerance level before admitting the call. And when the performance degrades, the system infers congestion and attempts to restrict or tear down lower priority calls. Yet another example could be an edge device intercepting RSVP setup messages and performing measurement, which would allow or deny the RSVP setup. There are a number of other variations to these examples, but essentially they all serve to infer network performance and perform Call Admission Control to the different flows.

Figure 3. End Device Measurement-based Solution



There are a few advantages to this solution. For one thing, the solution can be deployed without network changes as it is transparent to other network elements. It should handle any network topology, and it is fairly easy to deploy.

However, there are a number of drawbacks, too. First, this solution doesn't in any way protect or secure bandwidth reservation. It is merely a solution to infer the state of the network at that given point in time and adjust accordingly. In other words it is a reactive solution, reacting after congestion or quality degradation. This significantly affects user experience.

Secondly, it is neither accurate nor efficient in bandwidth utilization. For example, there will be occasions where the inference made could be misleading, especially in situations in which we have load-balancing paths in the network. The measurement path in this case could be different from the path the media eventually takes up. So a scenario could be sensing packets that are sent on a low utilization path while the actual media, through load balancing, is sent on a congested path.

Another reason for bandwidth utilization inefficiency is that the reactive behavior is completely distributed into the many end systems without any network control or central control. In case of a significant network failure affecting many sessions, many end systems will realize they are experiencing degradation. Since each endpoint needs to decide whether it should drop/reduce some sessions without being aware of exactly what is the overall level of congestion, how many endpoints are affected, and whether other endpoints will reduce their load, it is very difficult for a given endpoint to do the right thing so that collectively the endpoints all reduce their load to the exact new level tolerable by the network. A common endpoint behavior is to implement a recursive try-and-observe reduction; the endpoint will wait some random time, then reduce some flows and then observe. If there is still congestion, the endpoint will do the same one more time: reduce and observe. If the endpoints react too aggressively, then there will be an overshoot, and they will shed more sessions than necessary. If they react too slowly, then there will be an undershoot, and the congestion persists for a longer time until the recursive process reaches its conclusion. In summary, it is challenging to tune such a solution to simultaneously achieve reasonable bandwidth utilization efficiency and avoid persistent degradation (considering the variety of congestion scenarios to deal with).

In addition the solution tends to have an "oscillation" effect and have nondeterministic behaviors. At the time of congestion, when high-priority flows are getting affected, for example, video quality appearing to be jittery, low priority flows will be restricted access or throttled back when entering the core network, and the video quality eventually improves. Sensing that congestion is now alleviated, lower priority flows are now readmitted into the network thus causing the video quality to degrade yet again. And this cycle will repeat itself throughout. This oscillation could be mitigated by adopting a very conservative behavior whereby new flows are admitted very slowly and carefully, but this will unnecessarily block sessions that could have been accepted. Consider, for example, the case where a previously failed link comes back up again reintroducing lots of capacity in the system: one would want all new sessions to be admitted right away. In summary, it is very difficult (if not impossible) to tune such a reactive system to simultaneously achieve reasonable stability with yet reasonable efficiency.

Lastly, such solutions are not very amenable to strict policy enforcement. Reaction to congestion is left to the endpoints that have no visibility on conflicting sessions (for example, are there sessions contending for the same resource as my flow but with a higher priority?) and do not have visibility on the overall policy rules. Thus, enforcement of policy rules in such a solution is challenging. For example, in case of congestion detected by one endpoint, the endpoint cannot know whether it should drop low-priority sessions, low- and medium-priority sessions, or even some high-priority sessions. Again, the endpoint will then typically apply a recursive try-and-observe algorithm: start dropping some low-priority sessions; if there is still congestion, drop more low-priority sessions; if there is still congestion, drop medium-priority sessions; if there is still congestion, drop more medium-priority sessions; and if congestion still persists, drop high-priority sessions. This further increases the reaction time and thus during this period all sessions (including high-priority ones) experience degradation.

Because of these shortcomings, there is a strong incentive for defining an architecture that allows deployment of RSVP-based admission in multiple security domains networks. This is addressed in the remainder of this paper.

Building Blocks of the RSVP Solution

There are three major building blocks to solving the problem of RSVP deployment in a multiple security domains network. The good news is these building blocks are all based on standards or soon-to-be standards solutions, which should accelerate adoption, and they are as follows:

- RSVP Aggregation as specified in RFC 3175 and its enhancement in RFC 4860
- RSVP Signaling in Nested VPN as specified in RFC 4923
- Group keying and (Group Domain of Interpretation, as specified in RFC 3547) for RSVP message encryption support

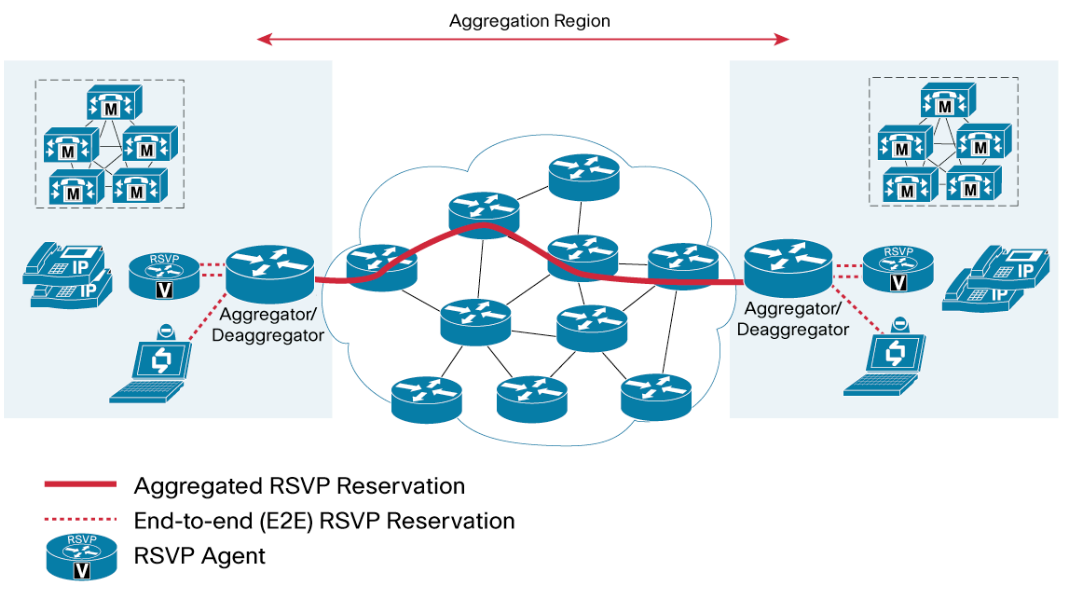
RSVP Aggregation

RSVP Aggregation Overview

RFC 3175 describes the use of a single RSVP reservation to aggregate other RSVP reservations across a transit routing region, in a manner conceptually similar to how virtual connections could be aggregated inside a virtual path in ATM (Asynchronous Transfer Mode) networks. It proposes a way to dynamically create the aggregate reservation, classify the traffic for which the aggregate reservation applies, determine how much bandwidth is needed to achieve the requirement, and recover the bandwidth when the subreservations are no longer required. It also contains recommendations concerning algorithms and policies for predictive reservations.

There is a notion of an aggregation region (see Figure 4), which really is a set of systems that should be isolated from (more numerous) individual reservations and should only handle (less numerous) aggregated reservations. At the edge of the aggregation region are routers capable of performing RSVP aggregation (and, of course, deaggregation). Communication interfaces fall into two categories with respect to an aggregation region; they are "exterior" to an aggregation region or they are "interior" to it. With respect to a given RSVP session, routers fall into one of three categories; they aggregate (aggregator), they deaggregate (deaggregator), or they are between an aggregator and a deaggregator (interior router). Sitting at the ingress and egress of this aggregation region are the aggregator and deaggregator, respectively. The function of an aggregator is to aggregate all the individual end-to-end (E2E) RSVP reservations destined for the same egress deaggregator as a single aggregated reservation. The aggregation router for this E2E flow is the first router that processes the E2E Path message as it enters the aggregation region (that is, the one that forwards the message from an exterior interface to an interior interface). The deaggregator router for this E2E flow is the last router to process the E2E Path message as it leaves the aggregation region (that is, the one that forwards the message from an interior interface to an exterior interface). An Interior router for this E2E flow is any router in the aggregation region that receives this message on an interior interface and forwards it to another interior interface. Interior routers perform neither aggregation nor deaggregation for this flow; they simply handle aggregate reservations. Note that by these definitions, a single router with a mix of interior and exterior interfaces may simultaneously act as an aggregator on some E2E flows, a deaggregator on other E2E flows, and an interior router on yet other flows.

Figure 4. RSVP Aggregation Overview



RSVP aggregation supports aggregation in a recursive manner so that aggregated reservations can themselves, in turn, be further aggregated, and so on, allowing a hierarchical aggregation deployment.

RSVP Aggregation Motivation

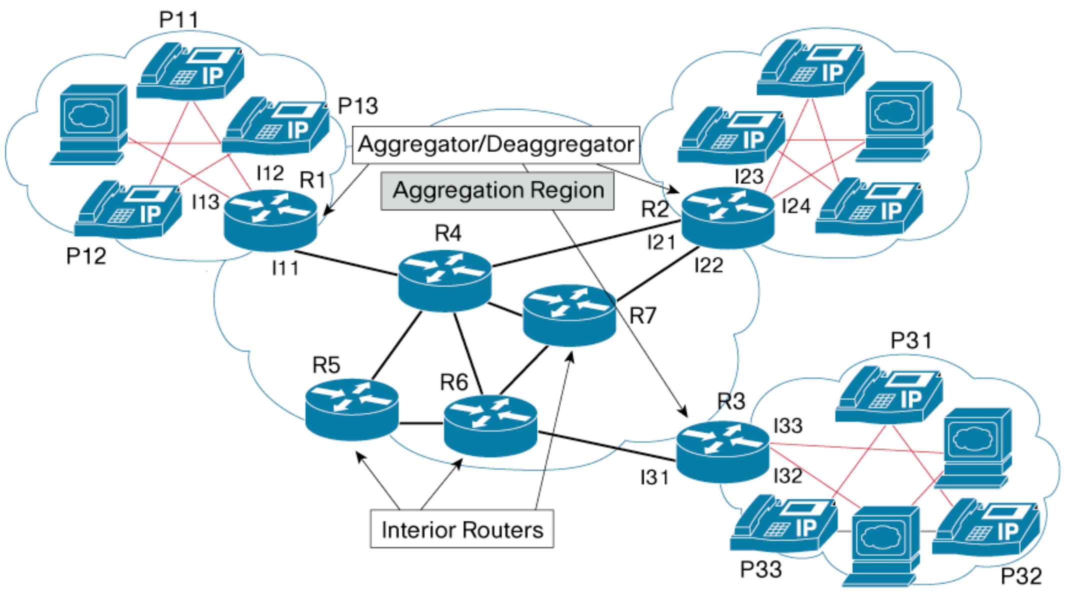
The original motivation for RSVP aggregation was to push RSVP scalability limits further. With regular RSVP, a router keeps states for every individual E2E reservation that transits through this router. This consumes memory on the router, and memory consumption grows linearly with the number of reservations crossing the router. In addition, per flow RSVP signaling in the control plane consumes route processing CPU cycles for establishment and teardown as well as for reservation maintenance through regular refresh. Thus it is not difficult to imagine that, typically, network backbone resources can't dedicate these kinds of per flow resources for thousands of video or voice streams.

The establishment of a smaller number of aggregate reservations on behalf of a larger number of E2E reservations yields the corresponding reduction in the amount of state to be stored and the amount of signaling messages exchanged in the aggregation region.

RSVP Aggregation Detailed Mechanism

Figure 5 shows an example RSVP reservation that illustrates the inner workings of RSVP aggregation. In this example IP Phone P11 attempts to setup a RSVP reservation toward IP Phone P32 with R1 and R3 acting as aggregator and deaggregator.

Figure 5. RSVP Aggregation Mechanism



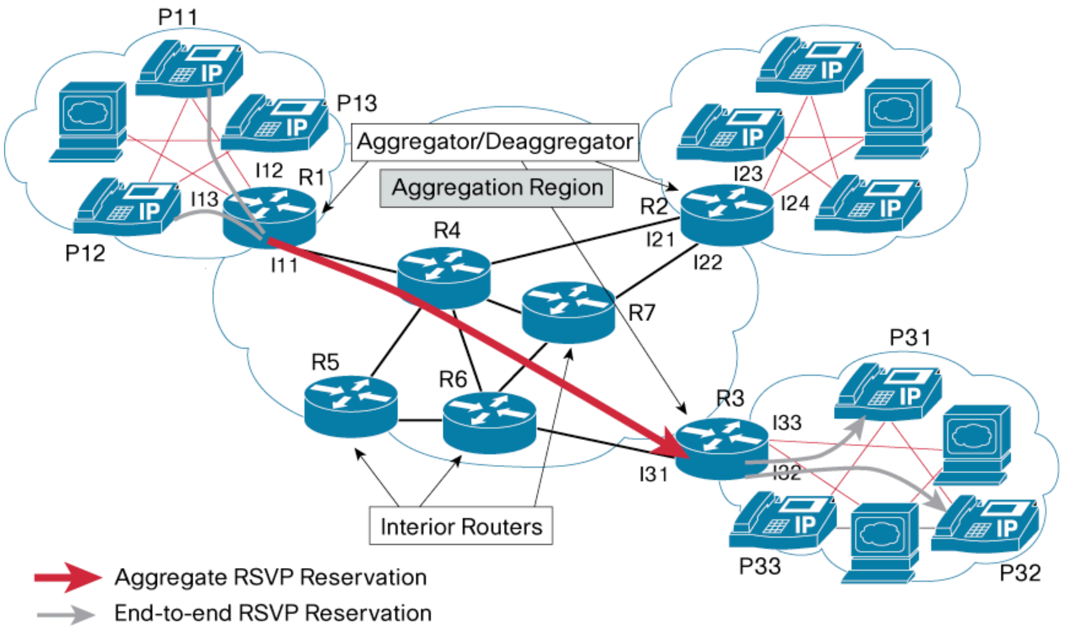
- Sender (P11)
  - Sends a Path message towards P31.
- Aggregator (R1)
  - Notices that this Path message is received on an exterior interface and is to be forwarded over an interior interface. Therefore R1 knows it needs to behave as an aggregator for this E2E reservation.
  - Processes the Path message and, when forwarding it, turns the protocol number from RSVP (46) to RSVP-E2E-IGNORE (134). This is so that all the routers along the reservation path in the aggregation region do not keep state for this E2E reservation.
- Interior routers (R4 and R7)
  - Will see router alert and thus intercept the Path message

    - Will notice that (i) the protocol is RSVP-E2E-IGNORE and (ii) both the Path message's inbound and outbound interfaces are configured as an interior interface
    - Will just forward the Path message without any further processing
- Deaggregator (R3)
    - Will see router alert and thus intercept the Path message.
    - Will notice that the protocol is RSVP-E2E-IGNORE and the Path message's inbound interface is interior while it is to be forwarded over an exterior interface.
    - Concludes that it needs to behave as the deaggregator for this E2E reservation.
    - Based on the mapping rule configured, knows that this E2E reservation is to be mapped on an aggregate reservation with (in this example) DSCP=EF.
    - Since there is currently no aggregate reservation from R1 (which appears as the PHOP in the E2E Path) to R3, R3 will trigger establishment of this aggregate reservation by sending an E2E PathError with Error=New-Aggregate-Needed and DCLASS=EF.
    - R3 will then forward the E2E Path message towards P31, restoring the protocol number to RSVP.
    - Optionally, the E2E Path message can be delayed until the Aggregate Path message is received.
- On receipt of the E2E PathError, R1 will generate an Aggregate Path message towards R3 (with DSCP=EF in the Session Object).
- R4 and R7 process the Aggregate Path and establish the corresponding Path states.
- On receipt of the Aggregate Path, R3 will respond with an Aggregate Resv whose FLOWSPEC is constructed (say) from per DSCP/per aggregator configuration commands on R3.
- R7, R4, and R1 will successively receive the aggregate reservation and perform admission control of this aggregate reservation over their egress interface bandwidth.
- On receipt of the E2E Resv from P31, R3 will perform admission control of the E2E Resv over the aggregate reservation, and if the aggregate reservation has sufficient bandwidth, R3 will accept it. R3 will then send the E2E Resv towards R1, including a DCLASS=EF. If the aggregate has insufficient bandwidth, R3 will send an E2E ResvError to P31.
- On receipt of the E2E Resv, R1 removes the DCLASS and forwards the Resv upstream.

If another E2E reservation needs to be established, then a similar process happens except that the aggregate reservation would already be in place, so that all the steps involved in establishing the aggregate reservation are skipped.

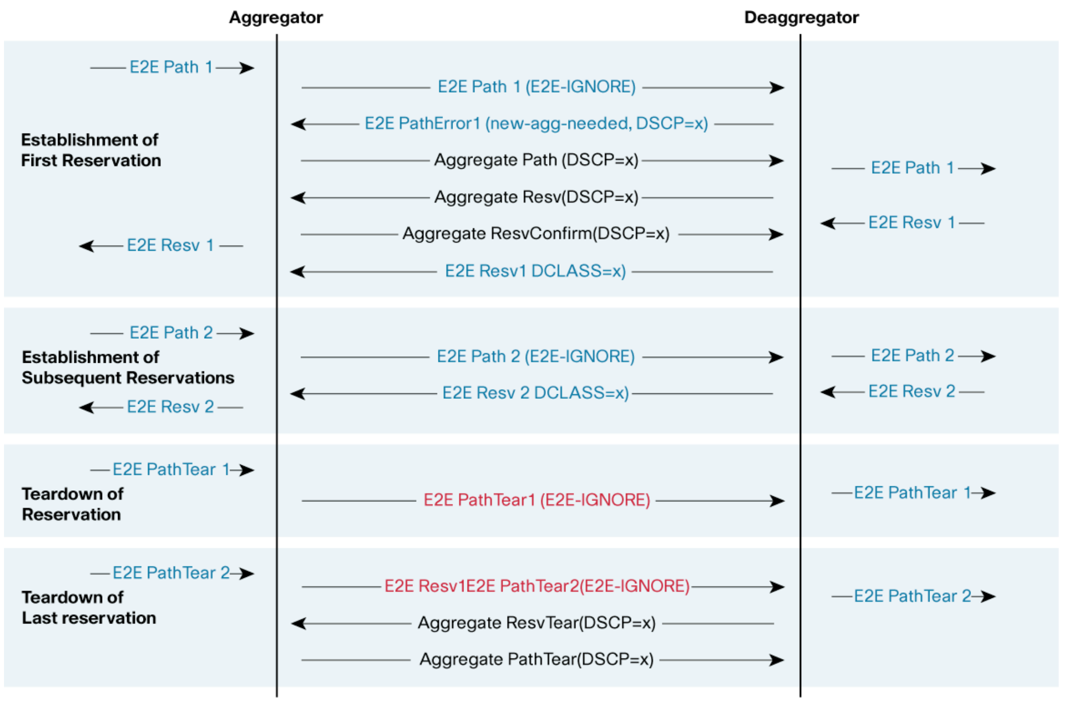
Figure 6 shows the end result of the RSVP aggregation process.

Figure 6. End Result of RSVP Aggregation Process



The detailed call flow of the above mechanism is depicted in Figure 7.

Figure 7. RSVP Aggregation Detailed Call Flow



Generic Aggregate RSVP Reservation

RFC 3175 defines aggregate RSVP reservations allowing resources to be reserved in a Diffserv network for a given per hop behavior (PHB) or a given set of PHBs from a given source (for example, the aggregator) to a given destination (for example, the deaggregator). Per hop behavior (PHB) refers to packets are classified and marked to receive a particular forwarding treatment.

However, from a given source IP address to a given IP destination address, only a single RSVP aggregate reservation can be established for a given PHB (or given set of PHBs). Situations have since been identified where multiple such aggregate reservations are needed for the same set of source IP address, destination IP address, and PHB (or set of PHBs). One example is where E2E reservations using different preemption priorities need to be aggregated through a Diffserv cloud using the same PHB. Using multiple aggregate reservations for the same PHB allows enforcement by admission control of the different preemption priorities at every hop within the aggregation region. In turn, this allows finer-grain management of the Diffserv resources, and in periods of resource shortage, this allows establishment of a larger number of E2E reservations with higher preemption priorities. To that end, RFC 4860 provides a small enhancement to RFC 3175 by defining a more flexible type of aggregate RSVP reservation, referred to as generic aggregate reservation. Multiple such generic aggregate reservations can be established for a given PHB (or set of PHBs) from a given source IP address to a given destination IP address. The generic aggregate reservations can be used by aggregators/deaggregators when aggregating end-to-end RSVP reservations. Note that the generic aggregate reservations could also be used natively by end systems that need to establish reservation for multiple sessions among each other.

Figure 8. Generic Aggregate RSVP Reservation

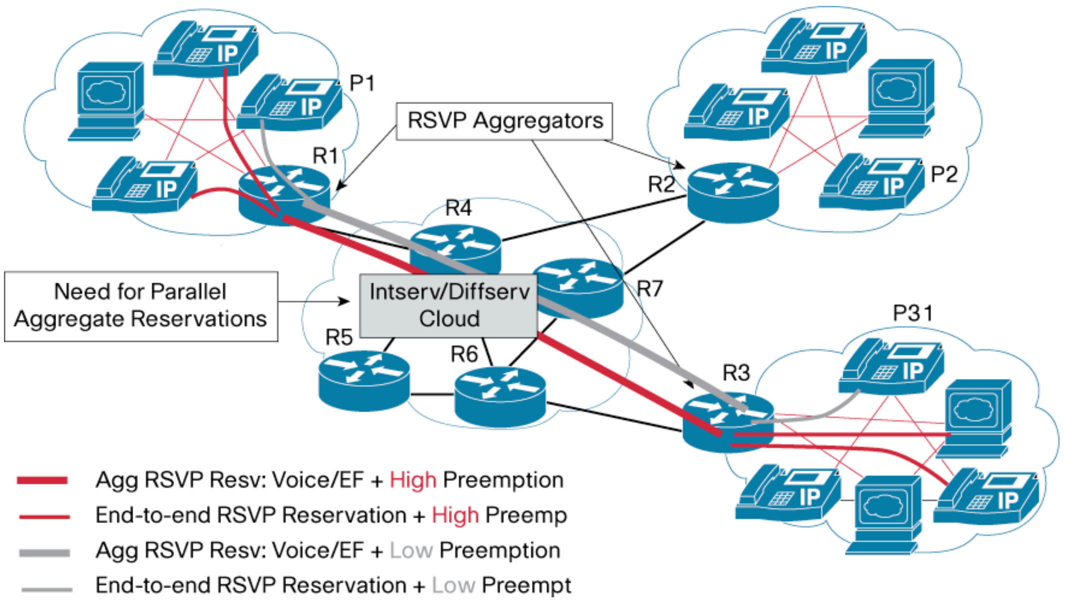

P1
RSVP Aggregators
R1
R2
P2
R4
Need for Parallel Aggregate Reservations
Intserv/Diffserv Cloud
R7
R5
R6
P31
R3
Agg RSVP Resv: Voice/EF + High Preemption
End-to-end RSVP Reservation + High Preemp
Agg RSVP Resv: Voice/EF + Low Preemption
End-to-end RSVP Reservation + Low Preempt

Figure 8 depicts an example involving two generic aggregate RSVP reservations for the same PHB from the same aggregator to the same deaggregator. The two generic aggregate RSVP reservations have different RSVP preemption priorities (high and low). In the event of congestion in the network path, the lower priority aggregate reservation might either be requested to reduce its bandwidth requirement or be terminated entirely, while the higher preemption priority aggregate reservation will be maintained.

Note that it is possible to support E2E reservation with different preemption priorities even when a single aggregate reservation is used (for a given PHB) from a given aggregator to a given deaggregator (and therefore without RFC 4860). This can be achieved by enforcing preemption priorities on the aggregator/deaggregator. However, this mandates that bandwidth reduction mechanisms (see the next section) be used in the core and cannot always maximize the number of higher preemption sessions that can be maintained by the network.

RSVP Reservation Bandwidth Reduction

RFC 4495 defines extensions to RSVP that allow the network, when justified, to reduce the bandwidth associated with a given reservation already in place. When used for aggregate reservations, this allows the network (when faced with a resource shortage) to reduce aggregate reservations instead of simply dropping them. At a system level, this allows better distribution of resources; for example, in the presence of congestion, it may be preferable to reduce the allowed communication level across all sites (in turn allowing all high-priority sessions to be maintained) rather than to maintain unaffected communications between some sites and block all communications across other sites (including high-priority sessions). This also allows the network to adjust a new reservation so it can reserve the maximum bandwidth left on that path rather than simply reject the reservation upfront because it requested more than available.

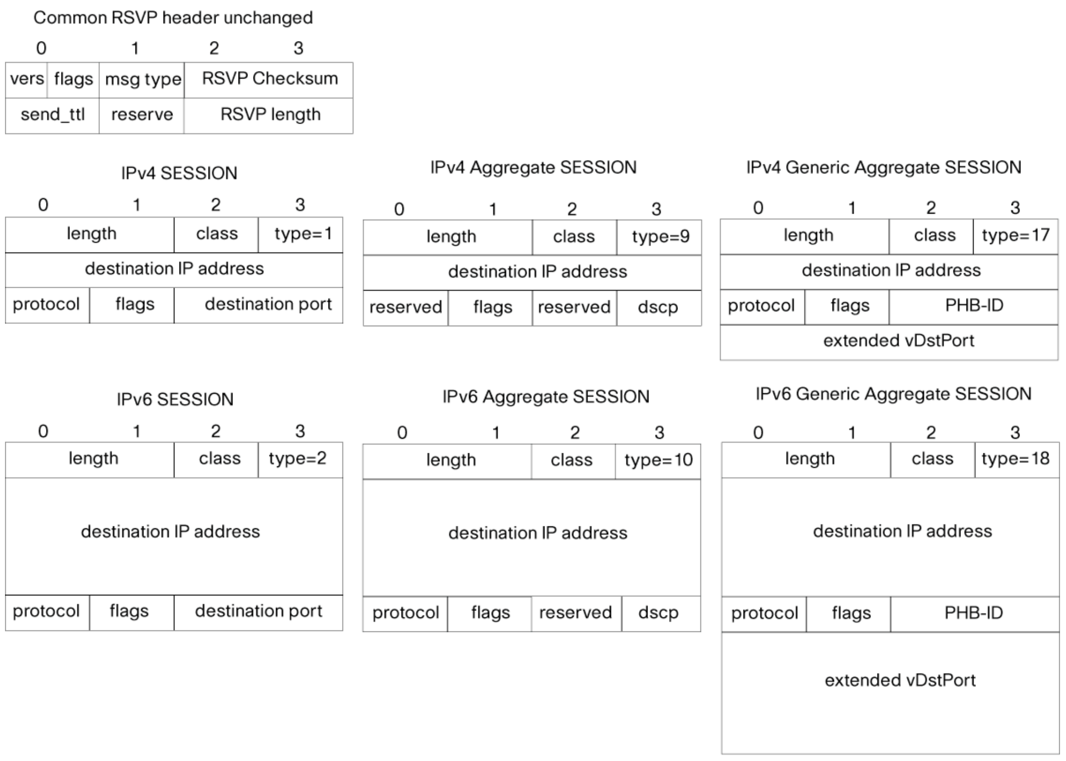
RSVP Aggregation over RSVP-TE MPLS Tunnels

RFC 3175 defines aggregation of RSVP reservations over aggregate IP reservations. RFC 4860 provides a variation by defining RSVP aggregation over a slightly different form of aggregate reservation: generic aggregate IP reservations. RFC 4804 specifies yet another variation whereby RSVP reservations are aggregated over yet another form of aggregate reservations: MPLS TE tunnels established through RSVP-TE. This is an attractive option where MPLS traffic engineering is deployed in the core: it combines the benefits of MPLS transport (fast reroute, constraint-based routing, bandwidth reservation) and the benefit of end-to-end RSVP admission control.

RSVP Aggregation Message Format

Figure 9 provides an illustration of how the message format has been modified to accommodate the capabilities provided by RSVP aggregation (including generic aggregate RSVP). The common header for RSVP has not changed; however, the Session object has a new format. There are couple salient points to note; in RSVP aggregate, the destination port field is no longer used but replaced by DSCP, and the generic aggregate includes a virtual destination port to accommodate the establishment of multiple aggregate sessions between the same pair of aggregator and deaggregator.

Figure 9. Message Format for RSVP Aggregation



Why RSVP Aggregation for Multiple Security Domains Network?

In a multiple security domains network, there is a strong security boundary between the red networks and the black network. In particular, individual endpoint addresses are hidden in the black core, and all traffic appears as coming from, and going to, a security gateway. The RSVP aggregation architecture fits very naturally in that model since it effectively “hides” E2E reservations from black core devices, only exposing aggregate reservations coming from the aggregator and terminating in the deaggregator.

Using aggregate RSVP reservations through the black network hides all detailed information. It helps ensure that passive monitoring of the traffic would not yield any sensitive information: the identity of endpoints participating in reservation signaling is hidden, and the amount of reservation from endpoints (or even for a given application flow) are also hidden. Adversaries intercepting RSVP aggregation messages would not be able to tell who the endpoints are and how many active calls are in session. Thus RSVP aggregation, in conclusion, is not only a step in the right direction but is also a necessary first step when it comes to information assurance.

Note also that the hierarchical aggregation supported by RSVP aggregation matches well with multiple hierarchical levels of security commonly deployed in a multiple security domains network.

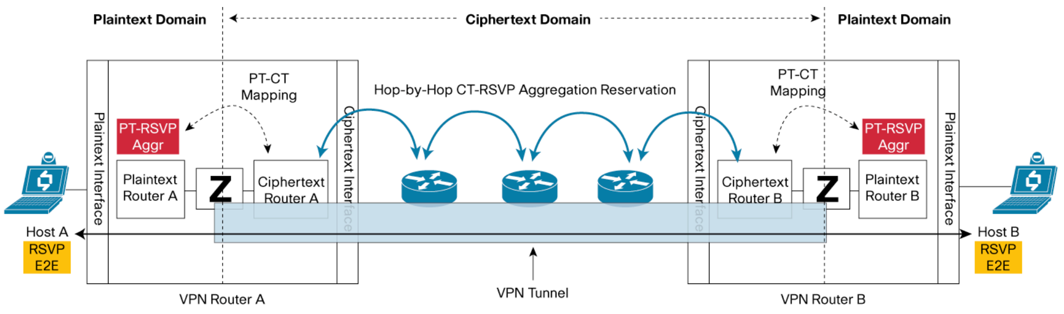
RSVP Signaling in Nested VPN as Specified in RFC 4923

Thus far, it would appear that a potential solution is as depicted in Figure 10. A VPN router is basically composed of a plaintext router, an encryption/decryption unit, and a ciphertext router. A plaintext router is responsible for establishing IP reachability in the plaintext domain (including over the VPN tunnels), whereas a ciphertext router is responsible for establishing IP reachability in the ciphertext domain. The plaintext routers below also function as aggregators/deaggregators. It is important to note that a VPN router has interfaces in two domains—plaintext and ciphertext.

To walk through an example of the potential solution, let's have host A send an RSVP Path message to host B. Please also refer to Figure 10.

- Plaintext-Router-A sends an RSVP Path message in an encrypted tunnel to VPN router B.
- Ciphertext routers in the ciphertext domain carry the encrypted RSVP Path message (like any other encrypted data message).
- Plaintext-Router-B processes the decrypted RSVP Path message and sends an E2E PathErr (New-Aggregate-Needed) message to Plaintext-Router-A in the encrypted tunnel.
- Plaintext-Router-B forwards a decrypted plaintext RSVP Path message to host B.
- Plaintext-Router-A receives E2E PathErr (from the tunnel) and generates a plaintext Aggregate Path message. It then sends a trigger to Ciphertext-Router-A to generate the corresponding ciphertext Aggregated Path message for the ciphertext domain. Note here that proper mapping and translation have to take place in both VPN router A and B between plaintext and ciphertext routers; otherwise, the aggregate RSVP Path message will not be forwarded properly.
- The ciphertext aggregate RSVP message travels through ciphertext routers in the ciphertext domain. This helps ensure that RSVP-capable ciphertext routers reserve the required resources for a plaintext end-to-end reservation. Subsequent mechanisms, such as preemption or the increase and decrease of resources, may be applied to these reservations. It is important to clarify that the ciphertext aggregate RSVP Path message is sent in the clear (unencrypted).
- Ciphertext-Router-B receives the ciphertext aggregate RSVP message and transforms the ciphertext aggregate RSVP into the plaintext aggregate RSVP message and sends it to Plaintext-Router-B.

Figure 10. Possible Solution Architecture



The above mechanism, however, presents a number of issues for certain organizations such as defense, intelligence, homeland security, and so on. For one thing, the solution architecture involves mapping of a plaintext aggregate RSVP message to a ciphertext aggregate RSVP message. There has to be a degree of control by the respective organizations to make sure that the mapping is done accurately, as these RSVP messages are crossing security domains. There needs to be a full assurance that no other messages apart from these aggregate RSVP messages are being propagated in ciphertext domain. Often policy may prevent knowledge of the ciphertext network addresses in the plaintext router. In addition these messages are sent in the clear (unencrypted). While aggregation does eliminate detailed information, it may be beneficial to have an option to encrypt these messages. For example, if the aggregate RSVP messages are sent in the clear, an observer could see reservations between some sites that are made ahead of the actual transmission of media packets. One could infer what sites are aware of "future" events, whatever they may be. Another good example is that one could see the actual preemption that is signaled inside an aggregate RSVP. Arguably that would allow an observer to infer which sites have more important traffic (which would otherwise not be detectable from traffic DSCP inspection). In subsequent paragraphs a solution to the aggregate RSVP mapping is presented, while aggregate RSVP encryption is addressed in the next section.

RFC 4923 describes a solution architecture for RSVP signaling in a nested VPN. A nested VPN is conceptually similar to a multiple security domains network. In RFC 4923 there is a description on the use of Network Guard to provide an additional path for reservation signaling between the plaintext and ciphertext domains. The Network Guard performs the following functions to help enable the flow of reservation signaling across the cryptographic domain:

- Transforms plaintext session identifiers into ciphertext session identifiers and vice versa in IP datagrams and RSVP objects (for example, IP addresses)
- Performs resource management of aggregated reservations (for example, factoring ciphertext encapsulation overhead into resources requested)
- Reads and writes configuration on the encrypt/decrypt units as necessary (for example, reads plaintext to ciphertext IP address mapping)

In addition, the plaintext and ciphertext routers must support a routing function or local interface that make sure that aggregated RSVP messages flow through the Network Guard.

Figure 11. Solution Architecture with Network Guard

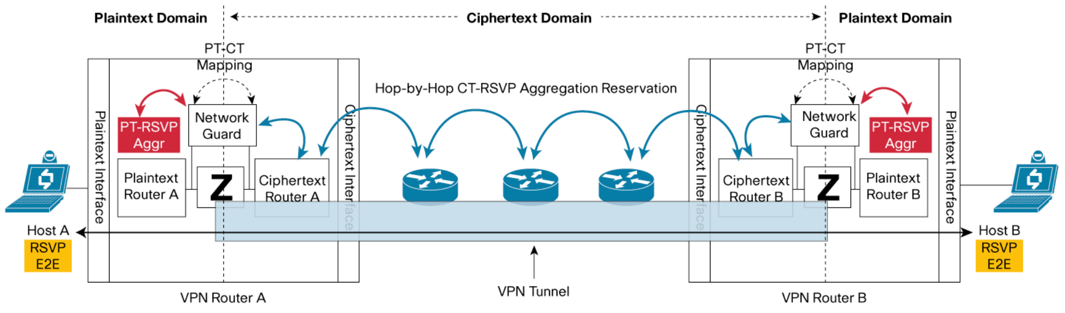


Figure 11 shows an example of Network Guard usage designed to address the issues of information assurance described earlier. Various organizations such as defense or homeland security can produce the Network Guard themselves and certify its functionalities in the overall solution. The implementation can be separate from the encryption/decryption unit depending on each country's policies and preferences.

A typical reservation set up in this case would follow these steps:

- Host A sends an RSVP Path message to host B.
- Plaintext-Router-A sends an RSVP Path message in an encrypted tunnel to VPN router B.
- Ciphertext routers carry the encrypted RSVP Path message (like any other encrypted data message).
- Plaintext-Router-B decrypts the RSVP Path message and sends an E2E PathErr (New-Aggregate-Needed) message to Plaintext-Router-A in the encrypted tunnel.
- Plaintext-Router-B forwards the decrypted plaintext RSVP Path message to host B.
- Plaintext-Router-A receives E2E PathErr and sends an aggregated RSVP Path message towards Plaintext-Router-B through the Network Guard.

- Network-Guard-A transforms the plaintext aggregate RSVP into the ciphertext aggregate RSVP message as described in Section 3.2.1 of RFC 4923 and sends it to Ciphertext-Router-A.
- The ciphertext aggregated RSVP message travels through ciphertext routers in the ciphertext domain to reserve the required resources for the plaintext end-to-end reservation.
- Ciphertext-Router-B receives the ciphertext aggregate RSVP message and sends it to Network-Guard-B.
- Network-Guard-B transforms the ciphertext aggregate RSVP into the plaintext aggregate RSVP message as described in Section 3.2.1 of RFC 4923 and sends it to Plaintext-Router-B.

The subsequent RSVP and aggregate RSVP signaling follows a similar flow, as described in detail in RFC 3175 and RFC 4860, to aggregate each plaintext reservation into a corresponding ciphertext reservation.

Group Domain of Interpretation as Specified in RFC 3547 for RSVP Message Encryption Support

As for security in RSVP messages, the current solution (RFC 2747 and RFC 3097) addresses message authentication, integrity protection, and replay protection. Specification for message confidentiality doesn't exist today. This is because confidentiality wasn't considered to be a security requirement for RSVP. This assumption does not hold in defense, homeland security, or intelligence networks and increasingly in enterprises who want to protect billing data, network usage patterns, or network configurations as well as users' identities from eavesdropping and traffic analysis. Confidentiality also has the potential of thwarting certain attacks. As the saying goes, "You can't attack what you can't see." Part of the reasons for not having a confidentiality specification for RSVP is also due to the challenges of applying IP Security (IPsec) encryption in this environment. There are issues to be dealt with in regard to key types and key provisioning methods, which are elaborated below.

Key Types

There are three key types that can be potentially used for RSVP message encryption support: interface-based, neighbor-based, and group keys. The following gives an overview of each key type and each type's feasibility for RSVP message encryption support.

Interface-based Keys

Most current RSVP authentication implementations support interface-based RSVP keys. When the interface is point to point (and therefore an RSVP router has only a single RSVP neighbor on each interface), this is similar to neighbor-based keys in the sense that a different key is used for each neighbor. However, when the interface is multipoint, all RSVP speakers on a given subnet have to share the same key in this model, which makes it unsuitable for deployment scenarios where different trust groups share a subnet, for example, Internet exchange points. In such a case, neighbor-based keys are required.

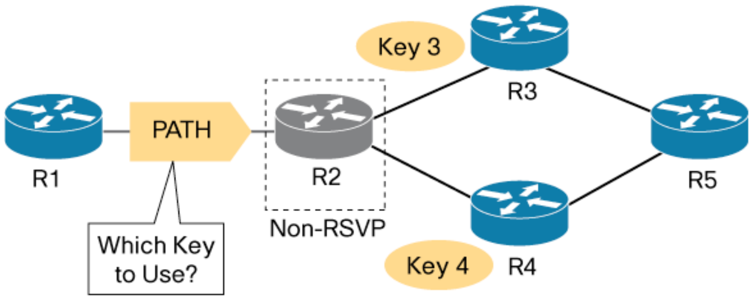
Neighbor-based Keys

In this model, an RSVP key is bound to an interface plus a neighbor on that interface. It allows the distinction of different trust groups on a single subnet. (Assuming that Layer 2 security is correctly implemented to prevent Layer 2 attacks.)

Group Keys

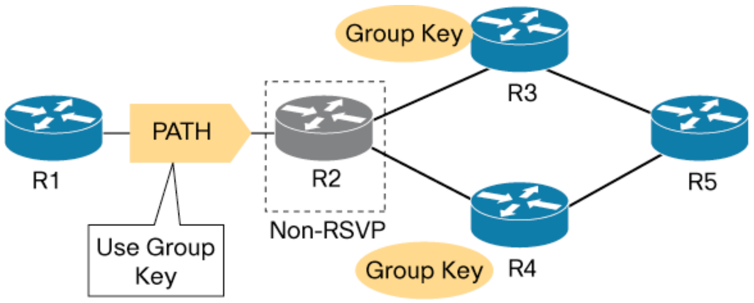
Here, all members of a group of RSVP nodes share the same key. This implies that a node uses the same key regardless of the next RSVP hop that will process the message (within the group of nodes sharing the particular key). It also implies that a node will use the same key on the receiving as on the sending side (when exchanging RSVP messages within the group).

Figure 12. Issue with Interface or per Neighbor Keying



In the presence of a non-RSVP-aware router in the path from sender to receiver, the non-RSVP-aware node will just ignore the RSVP message and transparently pass it along to the next node. So in Figure 12, R2 will ignore all RSVP messages and pass them along unchanged. Some RSVP messages (Path, PathTear, and ResvConf) are addressed to the (unicast or multicast) destination address and not to the next RSVP node along the path. Therefore with a non-RSVP hop in the path, with some RSVP messages such as the Path message or PathTear message, an RSVP router does not know the RSVP next hop for that message at the time of forwarding it. In fact, part of the role of a Path message is precisely to discover the RSVP next hop (and to dynamically rediscover it during a routing change). For example, in Figure 12, R1 knows that the next IP hop for a Path message addressed to the receiver is R2, but it does not necessarily know if the RSVP next hop is R3 or R4. Per interface and per neighbor keys will pose an issue here, as without knowing the next RSVP hop, R1 does not know which security association to use when forwarding the Path message. Note that other RSVP messages such as Resv, ResvTear, PathErr, and ResvErr do not have such a problem, as the RSVP next hop address is always used in forwarding the messages; however, a solution needs to be in place to cater for all RSVP message types.

Figure 13. RSVP Encryption with Group Keying



By contrast, group keying will naturally work in the presence of non-RSVP routers. In Figure 13, with group keying, R1 would use the group key to encrypt a Path message addressed to the receiver and would then forward it to R2. Being a non-RSVP node, R2 will ignore and forward the Path message to R3 or R4 depending on the cost metric as determined by routing. Whether it is R3 or R4, the RSVP router that receives the Path message will be able to decrypt it successfully with the group key.

Thus, group keying has the benefit of allowing activation of RSVP authentication and RSVP confidentiality mechanisms even in the presence of non-RSVP hops.

Subverted RSVP Nodes

A subverted node is defined here as an untrusted node, for example because an intruder has gained control over it. Since RSVP message authentication, encryption, replay protection, and integrity check are performed hop by hop and not end to end, a subverted node in the path breaks the chain of trust. This is to a large extent independent of the type of keying used. For interface or per neighbor keying, the subverted node can now introduce fake messages to its neighbors. This can be used in a variety of ways, for example by changing the receiver address in the Path message or by generating fake Path messages. This allows path states to be created on every RSVP router along any arbitrary path through the RSVP domain. That in itself could result in a form of denial of service by allowing exhaustion of some router resources (for example, memory). The subverted node could also generate fake Resv messages upstream corresponding to valid Path states. In doing so, the subverted node can reserve excessive amounts of bandwidth, thereby possibly performing a denial of service attack.

Group keying allows the additional abuse of sending fake RSVP messages to any node in the RSVP domain, not just adjacent RSVP nodes. However, in practice this can be achieved to a large extent also with per neighbor or interface keys, as discussed above. Therefore the impact of subverted nodes on the path is comparable, independently, whether per interface, per neighbor, or group keys are used.

Key Provisioning Methods

- Static key

  The simplest way to implement RSVP authentication is to use static, preconfigured keys. Static keying can be used with interface-based keys, neighbor-based keys, or group keys. However, such static key provisioning is expensive on the operational side, since no secure automated mechanism can be used, and initial provisioning as well as key updates require configuration. This method is therefore mostly useful for small deployments, where key changes can be carried out manually, or for deployments with automated configuration tools that support key changes. Static key provisioning is therefore not an ideal model in a large network.

- Per neighbor key negotiation

  To avoid the problem of manual key provisioning and updates in static key deployments, key negotiation between RSVP neighbors could be used. Key negotiation could be used to derive either interface- or neighbor-based keys. However, existing key negotiation protocols such as IKEv1 RFC 2409 or IKEv2 RFC 4306 may not be appropriate in all environments because of the relative complexity of the protocols and related operations.

- Dynamic key distribution using GDOI

  GDOI (Group Domain of Interpretation) describes a mechanism as specified in RFC 3547 to distribute to group members a group key that can be a group of RSVP nodes. In this model, a group controller/key server (GCKS) authenticates each of the RSVP nodes independently and then distributes a group key and security policy to the entire group. The GDOI protocol is run between a group member and GCKS, which establishes security associations among authorized group members. GCKS generates and pushes new IPsec keys (rekey) and policy to the routers when necessary. GDOI provides for efficient rekey of a group through the use of IP multicast messages that are received by all group members. Rekey messages can also cause group members to be ejected from the group. GDOI as per RFC 3547 supports dynamic key distribution for RSVP encryption. Where only RSVP authentication is required, the extensions to GDOI for RSVP authentication specified to that effect in draft-weis-gdoi-for-rsvp can be used.

GDOI Solution

Recall for the moment that the first solution building block is to aggregate RSVP reservation, as aggregation eliminates detailed information. To fulfill the requirement of information assurance, a Network Guard is proposed as the second solution building block to make sure that proper translation takes place between plaintext aggregate messages and ciphertext aggregate messages. The design and verification of a Network Guard functionality is under full control of an individual country's relevant organizations. As these ciphertext aggregate messages are sent in the clear, a third and final solution building block in the form of group key and GDOI will be used to provide encryption support to these messages.

Figure 14. Solution Architecture with GDOI

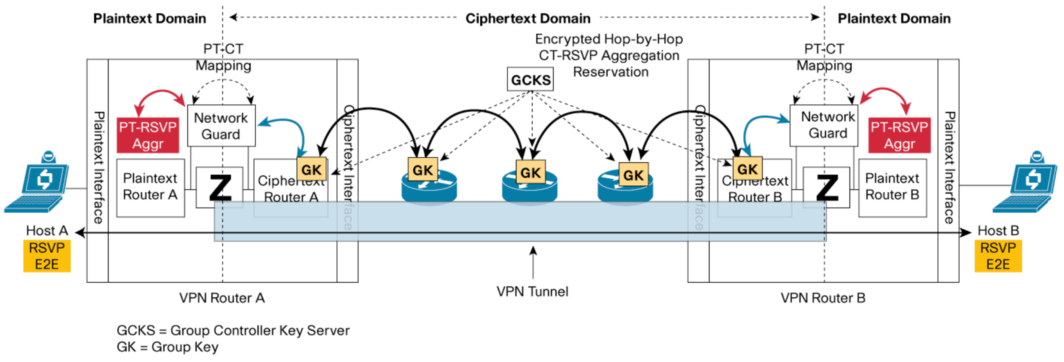
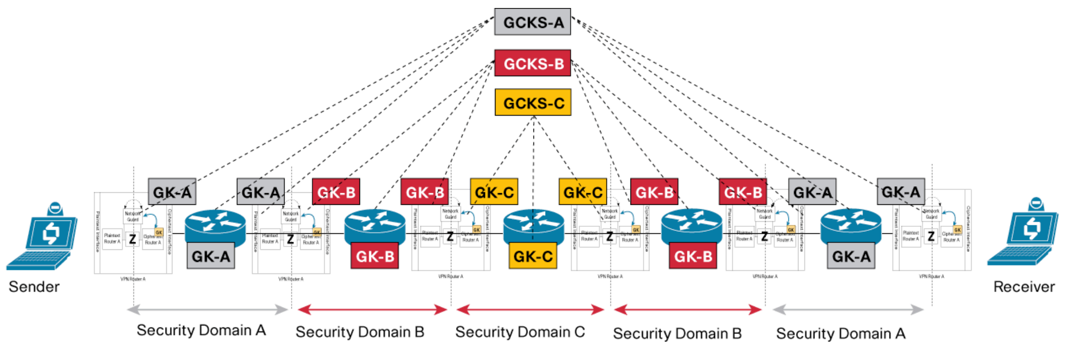


Figure 14 depicts the proposed solution using GDOI. All the ciphertext routers (A, B, and everything else in between) basically belong to a security domain that is served by a GCKS. They use a group key to encrypt/decrypt ciphertext aggregate messages. Since the aggregate RSVP messages originate from the plaintext domain, one could argue that in the event the ciphertext domain uses a lower assurance crypto to encrypt these messages, this may pose an issue. The beauty of aggregation, as mentioned, is such that it eliminates or "dilutes" detailed information. A diluted set of information has lesser sensitivity, and with that the security requirement can be lowered accordingly. And the same principle applies when information goes through more aggregation levels; that is, it becomes less and less sensitive.

Figure 15 shows a three-level aggregation with security domain A, B, and C. Each security domain is served with a GCKS that can be managed by administrators for that domain. As RSVP messages go from sender (left) to receiver (right), they will undergo three aggregation levels in VPN-A1, VPN-B1, and VPN-C1, and these RSVP messages get deaggregated at VPN-C2, VPN-B2, and VPN-A2 (in that order).

Figure 15. Three-Level RSVP Aggregation and Deaggregation



Conclusion

This paper presents an architecture allowing deployment of RSVP within a multiple security domains network commonly deployed by organizations such as defense, homeland security, and intelligence agencies worldwide. This architecture combines the following building blocks:

- RSVP aggregation as specified in RFC 3175 and its enhancement in RFC 4860
- RSVP signaling in nested VPN as specified in RFC 4923
- Group keying and extensions to GDOI (as specified in RFC 3547) for RSVP message authentication/encryption support

This architecture allows a multiple security domains network to take advantage of the benefits of end-to-end RSVP reservation and admission control such as strict QoS guarantees, deterministic performance, application-aware and user-aware policy enforcement, and efficient bandwidth utilization. At the same time, the architecture achieves this without compromising the levels of security assurance demanded by users in this environment.

For More Information

Learn more about deploying RSVP in multiple security domains networks by contacting RSVP-MSDN@cisco.com.

Other resources:

- Resource-Reservation Protocol (RSVP)

  http://www.cisco.com/en/US/docs/internetworking/technology/handbook/RSVP.html - wp1023211
- RSVP Aggregation and IOS Configuration Guide
  http://www.cisco.com/en/US/docs/ios/qos/configuration/guide/qos_rsvp_agg.html
- Group Encrypted Transport VPN Security Analysis
  http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6586/ps6635/ps7180/white_paper_c11-471053.html
- RFC 2205—Resource ReSerVation Protocol (RSVP)

  http://www.ietf.org/rfc/rfc2205.txt?number=2205
- RFC 3175—Aggregation of RSVP for IPv4 and IPv6 Reservations

  http://www.ietf.org/rfc/rfc3175.txt?number=3175
- RFC 3547—The Group Domain of Interpretation

  http://www.ietf.org/rfc/rfc3547.txt?number=3547

- RFC 4495—RSVP Extension for the Reduction of Bandwidth of a Reservation Flow
  http://www.ietf.org/rfc/rfc4495.txt?number=4495
- RFC 4804—Aggregation of RSVP Reservations over MPLS TE/DS-TE Tunnels
  http://www.ietf.org/rfc/rfc4804.txt?number=4804
- RFC 4860—Generic Aggregate RSVP Reservations
  http://www.ietf.org/rfc/rfc4860.txt?number=4860
- RFC 4923—QoS Signaling in a Nested Virtual Private Network
  http://www.ietf.org/rfc/rfc4923.txt?number=4923