TEAS Working Group
Internet-Draft
Obsoletes: 2747 3097 (if approved)
Intended status: Standards Track
Expires: 30 October 2026

R. Atkinson
Consultant
T. Li
Juniper Networks

28 April 2026

RSVP Cryptographic Authentication, Version 2
draft-ietf-teas-rsvp-auth-v2-01

Abstract

This document provides an algorithm-independent description of the format and use of RSVP's INTEGRITY object. The RSVP INTEGRITY object is widely used to provide hop-by-hop integrity and authentication of RSVP messages, particularly in MPLS deployments using RSVP-TE. This document obsoletes both RFC2747 and RFC3097.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 30 October 2026.

Table of Contents

1. Introduction

The Resource ReSerVation Protocol RSVP [RFC2205] is a protocol for setting up distributed state in routers and hosts. It has two common uses in the deployed Internet. First, it is used to reserve resources to deploy Integrated Services. When used in this way, RSVP allows particular users to obtain preferential access to network resources, under the control of an admission control mechanism. Permission to make a reservation will depend upon the availability of the requested resources along the path of the data and satisfaction of policy rules. Second, it is used to create and manage MPLS Label Switched Paths (LSPs), possibly with resources being reserved on a

per-LSP basis.

To ensure the integrity of the admission control mechanism RSVP requires the ability to protect its messages against corruption and forgery. Where RSVP-TE is used to manage MPLS Label Switched Paths (LSPs), it is also important to mitigate the risk of unauthorized creation, modification, or deletion of a Label Switched Path.

This document defines a mechanism to protect RSVP messages in a hop-by-hop manner. When this mechanism is employed, the sending RSVP system transmits a cryptographic value in the Authentication Data field of the RSVP INTEGRITY object which is contained within each RSVP message. The cryptographic value is computed using information within an RSVP Security Association, which is precisely defined in Section 5.1. Hence, this mechanism can significantly reduce the security risks from forgery or modification of RSVP (including RSVP-TE) messages.

The INTEGRITY object of each RSVP message is also tagged with a one-time-use sequence number, which is described in more detail in Section 3. This helps the message receiver identify replayed RSVP messages and hence thwart replay attacks.

This mechanism does not provide confidentiality, since messages stay in the clear; however, the mechanism is also believed to be importable to and exportable from all countries, which would be impossible were a confidentiality mechanism included.

This document is a revision of [RFC2747]. The most important difference is that this document specifies an authentication mechanism in a manner which is independent of the cryptographic algorithm (e.g., MD5, SHA-256, SHA-3) used and the cryptographic mode (e.g., HMAC, GMAC, KMAC) used. This supports future evolution with a variety of other cryptographic algorithms and cryptographic modes without needing to change the RSVP Authentication mechanism. An algorithm-independent specification is important because historically all published cryptographic algorithms eventually become computationally weak or will have cryptographic flaws discovered.[DS-1981][RFC6151] Separately, different cryptographic algorithms will have different cryptologic and mathematical properties, which can mean that the cryptographic mode suitable for one algorithm is either unsuitable or less appropriate for some other cryptographic algorithm. As an example, while the HMAC construction [RFC2104] [NIST-HMAC] is sensible for Merkle-Damgard hash functions (e.g., SHA-1, SHA-2), the Keccak Message Authentication Code (KMAC) construction [NIST-KMAC] is preferable for the newer NIST SHA-3 hash function. [WAGNER] NIST also has defined the GCM Message Authentication Code (GMAC), which is another cryptographic authentication algorithm that might be used in the future. [NIST-GMAC]

This document only specifies the RSVP authentication mechanism and protocol and does not specify a particular cryptographic algorithm or cryptographic mode that MUST be implemented. Instead, this specification creates a new IANA Registry for the "RSVP Cryptographic Transform" within the existing RSVP registry group. This enables the set of mandatory, optional, and deprecated cryptographic mechanisms to be updated over time without needing to update or modify this document.

The RSVP checksum MAY be disabled (i.e., set to zero) when the INTEGRITY object is included in the RSVP message, as cryptographic authentication inherently provides a much stronger integrity check.

1.1. Terminology

Within this document, there are certain terms and concepts the reader should be familiar with.

First, this document uses the terms "sender" and "receiver" differently from [RFC2205]. They are used here to refer to systems that face each other across an RSVP hop, the "sender" being the system generating RSVP messages.

An "Authentication Key" is an unpredictable cryptographic key that is used in the calculation of the Authentication Data field of the RSVP Integrity Object. It is defined precisely in Section 5.1

The term "Cryptographic Transform" is the combination of a cryptographic algorithm (e.g., MD5, SHA-1, SHA-256), the length of the secret Authentication Key (e.g., 256 bits), and a cryptographic mode (e.g., HMAC, GMAC, KMAC). Each different Cryptographic Transform ought to be defined in its own RFC and provide a clear specification of how the Authentication Data field is calculated when that Cryptographic Transform is used in an RSVP Security Association.

The term "RSVP Security Association" and its contents are precisely defined in Section 5.1. It contains the Cryptographic Transform in use, the cryptographic key, and several other parameters.

1.2. Requirements Language

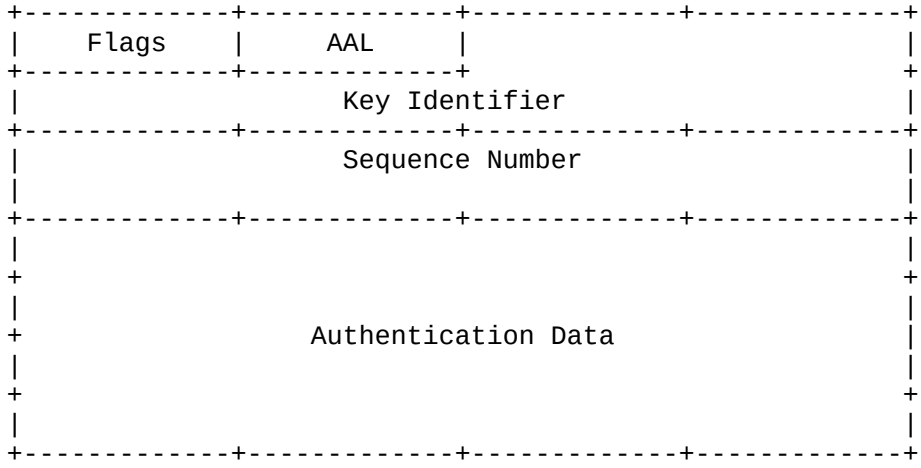
The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

2. INTEGRITY Object Format

An RSVP message consists of a sequence of "objects," which are type-length-value encoded fields having specific purposes. The information required for hop-by-hop integrity checking is carried in an INTEGRITY object. The same INTEGRITY object type is used for both IPv4 and IPv6.

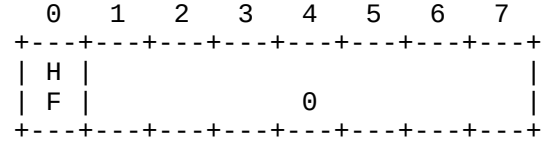
The INTEGRITY object has the following format:

Authentication Information INTEGRITY Object: Class = 4, C-Type = 1



* Flags: An 8-bit field with the following format:

Flags



Currently, only one flag (HF) is defined. The remaining flags are reserved for future use and MUST be set to 0.

* Bit 0: Handshake Flag (HF) concerns the integrity handshake mechanism (Section 4.3). Message senders willing to respond to integrity handshake messages SHOULD set this flag to 1 whereas those that will reject integrity handshake messages SHOULD set this to 0.

* Additional Authentication Length (AAL): This 8-bit field contains an unsigned integer. If this value is 0, the Authentication Data field contains 16 bytes. If this value is greater than 0, it specifies how many additional 4-byte increments (i.e., beyond the default 16 bytes) exist in the Authentication Data field. For example, a value of 1 indicates that the Authentication Data field is 20 bytes long (16 bytes of default plus one 4-byte addition). This counts in 4-byte increments so that 32-bit alignment is maintained within the RSVP message. If some future cryptographic transform has a result that is not 32-bit aligned, then the RFC specifying the cryptographic transform MUST document the length of the actual output and MUST also document that any trailing bytes after the length of the actual cryptographic result are padding.

* Key Identifier: An unsigned 48-bit number that MUST be unique for a given sender. Locally unique Key Identifiers can be generated using some combination of the address (IP or MAC or Logical Interface Handle (LIH)) of the sending interface and the key number. The combination of the Key Identifier and the sending system’s IP address uniquely identifies the security association (Section 5.1).

* Sequence Number: An unsigned 64-bit sequence number. Sequence Number values may be any monotonically increasing (modulo 2^64) sequence that provides the INTEGRITY object of each RSVP message with a unique tag for the associated key’s lifetime. Sequence number generation is specified below in Section 3.

* Authentication Data: This is an unsigned variable length field containing the cryptographic authentication data. The field length MUST be a multiple of 4 octets (i.e., 32 bits) long. If one knows the RSVP Cryptographic Transform used for a given RSVP packet, then one will know the correct length of this field. Given the combination of the Key Identifier and the Sender, one knows the RSVP Security Association, which includes the RSVP Cryptographic Transform.

2.1. Backward Compatibility

In [RFC2747], the INTEGRITY object contained a fixed 16-octet field for "Keyed Message Digest" for HMAC-MD5. This field is now renamed to "Authentication Data". Further, the second octet in the object was reserved and its contents were specified as 0. This field has now been redefined as the AAL field.

These changes should be fully backward compatible. A legacy implementation will continue to generate a 16-octet "Keyed Message Digest" and new implementations that expect HMAC-MD5 should continue

to expect 16 octets. Legacy implementations will continue to generate a reserved field containing 0. Similarly, an implementation conforming to this document that is generating HMAC-MD5 should continue to generate an Authentication Data field of 16 octets and an AAL field containing 0. These should be received by a legacy implementation without any differences noted.

3. Generating Sequence Numbers

In this section, we describe methods that could be chosen to generate sequence numbers for the INTEGRITY object of an RSVP message.

The sequence number field is chosen to be a 64-bit unsigned quantity. This should be large enough to avoid exhaustion over the key lifetime. For example, if a key lifetime is conservatively defined as one year, there would be enough sequence number values to send RSVP messages at an average rate of about 585 gigaMessages per second. A 32-bit sequence number would limit this average rate to about 136 messages per second.

As previously stated, two important properties MUST be satisfied by the generation procedure. The first property is that the sequence numbers are unique, or one-time, for the lifetime of the integrity key that is in current use. A receiver can use this property to distinguish unambiguously between a new and a replayed message. The second property is that the sequence numbers are generated in monotonically increasing order, modulo 2^64. This greatly reduces the amount of saved state.

It is desirable that RSVP Sequence Numbers not be trivially predictable. Therefore, the sequence numbers might not begin with "0", "1", or any other fixed number. At the start of an RSVP session, a receiver MUST handshake with the sender to get an initial sequence number. Since the starting sequence number might be arbitrarily large, the modulo operation described above accommodates sequence number rollover within the key’s lifetime. This initial sequence number selection solution draws from the approach in the updated specification for TCP [RFC9293].

This memo later discusses ways to relax the strictness of the in-order delivery of messages as well as techniques to generate monotonically increasing sequence numbers that are robust across sender failures and restarts.

The ability to generate unique monotonically increasing sequence numbers across a failure and restart implies some form of stable storage local to the device. Three sequence number generation procedures are described below.

3.1. Simple Sequence Numbers

The most straightforward approach is to generate a unique sequence number using a message counter. Each time a message is transmitted for a given key, the sequence number counter is incremented. The current value of this counter is continually or periodically saved to stable storage. After a restart, the counter is recovered using this stable storage. If the counter was saved periodically to stable storage, the count should be recovered by increasing the saved value to be larger than any possible value of the counter at the time of the failure. This can be computed by knowing the interval at which the counter was saved to stable storage and incrementing the stored value by that amount.

The periodicity of saving the sequence numbers need not be tied to time. This could also be implemented in terms of the usage of sequence numbers themselves. For example, an implementation could record the sequence number once every 1000 messages. On recovery, the implementation could recover the stored sequence number, advance it by 1000, and be reasonably assured that the new number is unique.

3.2. Sequence Numbers Based on a Real-Time Clock

Most devices will probably not have the capability to save sequence number counters to stable storage for each RSVP session.

A more universal solution is to base sequence numbers on the stable storage of a real-time clock. Many computing devices have a real-time clock module that includes stable storage for the clock. These modules generally include either some form of nonvolatile memory to retain clock information in the event of a power failure or have a small on-board battery to keep the clock running even when the device is not in use.

Also, many systems have deployed the Network Time Protocol (NTP) [RFC5905], the Precision Time Protocol (PTP) [IEEE-1588-2019] or a related ITU-T profile of the Precision Time Protocol [G.8265.1].

In this approach, we could use a Network Time Protocol (NTP) timestamp value or a Precision Time Protocol (PTP) timestamp value as the sequence number. The rollover period of an NTP timestamp is about 136 years, much longer than any reasonable lifetime for a key. In addition, the granularity of the NTP timestamp is fine enough to allow the generation of an RSVP message every 200 picoseconds for a given key. PTP timestamps have even finer granularity. Many real-time clock modules do not have the resolution of an NTP timestamp or PTP timestamp. In these cases, the least significant bits of the timestamp can be generated using a message counter, which is reset

every clock tick. For example, when the real-time clock provides a resolution of 1 second, the 32 least significant bits of the sequence number can be generated using a message counter. The remaining 32 bits are filled with the 32 least significant bits of the timestamp. Assuming that the recovery time after failure takes longer than one tick of the real-time clock, the message counter for the low-order bits can be safely reset to zero after a restart.

3.3. Sequence Numbers Based on a Network-Recovered Clock

If the device does not contain any stable storage of sequence number counters or of a real-time clock, it could recover the real-time clock from the network using either NTP, PTP, or the ITU-T profile of PTP. Once the clock has been recovered following a restart, the sequence number generation procedure would be identical to the procedure described above. To reduce the risk of forgery attacks and the risk of time-based RSVP attacks generally, deployments using NTP, PTP, or a different distributed clock protocol always SHOULD enable cryptographic authentication for time distribution.

4. Message Processing

Implementations MUST support the specification of RSVP Authentication on a per-interface basis. Implementations SHOULD also support the specification of RSVP Authentication on a per-peer basis.

4.1. Per-Interface Implementations

Implementations MUST allow the specification of the interfaces to be secured, for either sending messages, receiving them, or both. The sender must ensure that all RSVP messages sent on secured interfaces include an INTEGRITY object, generated using the appropriate Key. Receivers verify whether RSVP messages, except for the type "Integrity Challenge" (Section 4.3), arriving from a secured peer contain the INTEGRITY object. If the INTEGRITY object is absent, the receiver discards the message.

Security associations are simplex - the keys that an originating system uses to sign its messages MAY be different from the keys that its responder systems use to sign their messages back to the originator. Hence, each association corresponds to a unique sending system and one or more responding systems.

Each sender SHOULD have distinct security associations (and keys) per secured interface (or LIH). While administrators MAY configure all the routers and hosts on a subnet (or MAY, for that matter, all devices in their network) using a single security association, implementations MUST assume that each sender might send using a

distinct security association for each secured interface. At the sender, security association selection is based on the egress interface through which the message is sent. This selection MAY also include additional criteria, such as (a) the destination address (when sending the message unicast, over a broadcast LAN with a large number of hosts) or (b) user identities at the sender or receivers [RFC2752]. Finally, all intended message recipients need to participate in this security association. Route flaps in a non-RSVP cloud might cause messages for the same receiver to be sent on different interfaces at different times. In such cases, the receivers should participate in all possible security associations that might be selected for the interfaces through which the message might be sent.

Receivers select keys based on the Key Identifier and the sending system's IP address. The Key Identifier is included in the INTEGRITY object. The sending system's address can be obtained either from the RSVP_HOP object, or if that's not present (as is the case with PathErr and ResvConf messages) from the IP source address. The combination of the sending system's IP address and Key Identifier uniquely identifies the Security Association, including all of the data elements of a Security Association.

The integrity mechanism slightly modifies the processing rules for RSVP messages, both when including the INTEGRITY object in a message sent over a secured sending interface and when accepting a message received on a secured receiving interface. These modifications are detailed below.

4.1.1. Message Generation (Per-Interface)

For an RSVP message sent over a secured sending interface, the message is created as described in [RFC2205], with these exceptions:

1. The RSVP checksum field is set to zero. If required, an RSVP checksum can be calculated when the processing of the INTEGRITY object is complete.

2. The INTEGRITY object is inserted in the appropriate place, and its location in the message is remembered for later use.

3. The sending interface and other appropriate criteria (as described above) are used to determine the correct Security Association to use. The Security Association will specify the Cryptographic Algorithm, Cryptographic Mode, and Authentication Key to use.

4. The unused flags in the INTEGRITY object MUST be set to 0. The Additional Authentication Length (AAL) and the Handshake Flag (HF) should be set according to the rules specified in Section 2.

5. The sending sequence number MUST be updated to ensure a unique, monotonically increasing number. It is then placed in the Sequence Number field of the INTEGRITY object.

6. The Authentication Data field is set to zero.

7. The Key Identifier is placed into the INTEGRITY object.

8. Authentication Data for the message is computed over the message, using the Authentication Algorithm and Mode in conjunction with the Authentication Key.

9. The computed Authentication Data is written into the Authentication Data field of the INTEGRITY object.

4.1.2. Message Reception (Per-Interface)

When the message is received on a secured receiving interface and is not of the type "Integrity Challenge", it is processed in the following manner:

1. The RSVP checksum field is saved and the field is subsequently set to zero.

2. The Authentication Data field of the INTEGRITY object is saved and the field is subsequently set to zero.

3. The Key Identifier field and the sending system address determine the precise Security Association to use for the received message.

If the RSVP Security Association has expired AND a different RSVP Security Association is valid, then the packet MUST be discarded without cryptographic processing AND a security error SHOULD be logged in an implementation-specific manner (e.g., via SYSLOG or SNMP). Any such logging MUST be rate-limited to mitigate the potential for Denial of Service attacks.

If the RSVP Security Association has expired AND there is no RSVP Security Association valid at the time the RSVP packet was received, then the expired RSVP Security Association should be used to validate the received RSVP packet just as if the RSVP Security Association were not expired.

The RSVP Security Association is defined in Section 5.1. It includes the Cryptographic Transform, Authentication Key, and other parameters needed to validate the received RSVP packet.

1. A new Authentication Data value is calculated using the indicated Cryptographic Algorithm, Cryptographic Mode, and and Authentication Key.

2. If the calculated Authentication Data is not identical to the received Authentication Data, then the message MUST be discarded without further processing. A security error message SHOULD be logged (e.g., using SYSLOG or SNMP) if the received RSVP message is discarded for that reason, but any such error messages MUST be rate-limited to reduce risks from Denial of Service attacks.

3. If the message is of type "Integrity Response", verify that the CHALLENGE object identically matches the originated challenge. If it matches, save the sequence number in the INTEGRITY object as the largest sequence number received to date.

Otherwise, for all other RSVP Messages, the sequence number is validated to prevent replay attacks, and messages with invalid sequence numbers are ignored by the receiver. Validation is discussed in more detail in the following paragraphs.

When a message is accepted, the sequence number of that message SHOULD update a stored value corresponding to the largest sequence number received to date. In a naive implementation, each subsequent message would need to have a larger (modulo 2^64) sequence number to be accepted to reduce risks from replay attacks. However, this simple processing rule SHOULD be modified to tolerate limited out-of-order message delivery. For example, if several RSVP messages were sent in a burst (e.g., in a periodic refresh generated by a router, or as a result of a tear-down operation), then some of those RSVP messages might get reordered during transit and then the RSVP sequence numbers would not be received in a strictly increasing order.

An implementation SHOULD allow administrative configuration that sets the receiver’s tolerance to out-of-order message delivery. A simple approach would allow administrators to specify a received RSVP message window corresponding to the worst-case reordering behavior. For example, one might specify that packets reordered within a 32 message window would be accepted. If no reordering is allowed by policy, then the out-of-order received RSVP message window is set to one.

The receiver MUST store a list of all RSVP sequence numbers seen within the reordering window. A received RSVP sequence number is valid if it lies within the reordering window AND it is not a sequence number in the received sequence number list. Acceptance of a sequence number by an implementation requires adding that number to the received sequence number list. If the oldest sequence number within the window is received, then the window advances. A single message can cause the window to advance multiple times. Implementations MUST discard RSVP messages received with RSVP sequence numbers either (a) lying outside of the reordering window or (b) marked as already received in the RSVP received sequence number list.

When an "Integrity Challenge" message is received on a secured sending interface it is processed in the following manner:

1. An "Integrity Response" message is formed using the Challenge object received in the challenge message.

2. The message is sent back to the receiver, based on the source IP address of the challenge message, using the "Message Generation" steps outlined above. The selection of the Authentication Key and the hash algorithm to be used is determined by the key identifier supplied in the challenge message.

4.2. Per-Peer Implementations

Per-peer implementations follow the procedures in Section 4.1 but use the security associations defined for the specific peer.

4.2.1. Message Generation (Per-Peer)

Per-peer implementations follow the procedures in Section 4.1.1 for Message Generation, except using the security associations for the specific peer.

4.2.2. Message Reception (Per-Peer)

Per-peer implementations follow the procedures in Section 4.1.2 for Message Reception, except using the security associations for the specific peer.

4.3. Integrity Handshake at Restart or Initialization of the Receiver

To obtain the starting sequence number for a live Authentication Key, the receiver SHOULD initiate an integrity handshake with the sender. This Integrity Handshake consists of a receiver's Challenge and the sender's Response. The Integrity Handshake MAY either be initiated during restart or postponed until a message signed with that key arrives.

To ensure interoperability, and mindful that the Integrity Handshake might be essential to synchronize understanding of the starting sequence number, implementations of this specification MUST implement this Integrity Handshake capability.

Implementations of the older [RFC2747] RSVP Cryptographic Authentication specification might not have implemented the Integrity Handshake. The Handshake Flag (HF) enables backwards compatibility with those legacy implementations by allowing implementations to indicate they do not implement the Integrity Handshake mechanism. An implementation that does not implement the Integrity Handshake MUST set the HF flag to 0. Message senders that implement the integrity handshake MUST set the HF flag to 1. Receivers SHOULD NOT attempt to handshake with senders whose INTEGRITY object has HF = 0.

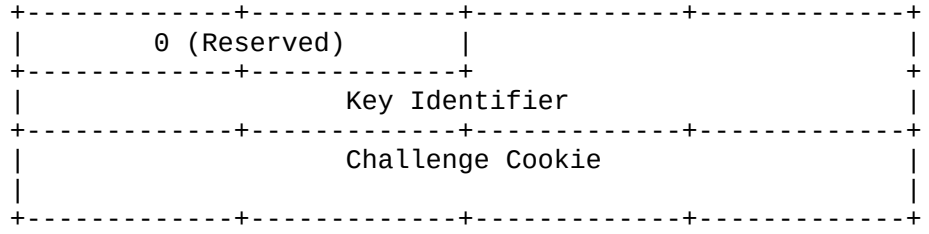
Once the receiver initiates an Integrity Handshake for a particular RSVP Security Association, it identifies the sender using the sending system's address configured in the corresponding RSVP Security Association. The receiver then sends an RSVP Integrity Challenge message to the sender. This message contains the Key Identifier to identify the sender's key and MUST have a unique, unpredictable challenge cookie to prevent guessing. See Section 2.5.3 of [RFC2408]. One implementation option is to have the cookie be a cryptographic hash of an unpredictable local secret, an unpredictable random number (see [RFC8937] and [NIST-ENTROPY]), and a timestamp to provide uniqueness (also see Section 3.2).

An RSVP Integrity Challenge message will carry a message type of 25. The message format is as follows:

```
<Integrity Challenge message> ::= <Common Header> <CHALLENGE>
```

The CHALLENGE object has the following format:

CHALLENGE Object: Class = 64, C-Type = 1



The sender accepts the "Integrity Challenge" without doing an integrity check. It returns an RSVP "Integrity Response" message that contains the original CHALLENGE object. It also includes an INTEGRITY object, signed with the key specified by the Key Identifier included in the "Integrity Challenge".

An RSVP Integrity Response message will carry a message type of 26. The message format is as follows:

```
<Integrity Response message> ::= <Common Header> <INTEGRITY>
                      <CHALLENGE>
```

The "Integrity Response" message is accepted by the receiver (challenger) only if the returned CHALLENGE object matches the one sent in the "Integrity Challenge" message. This prevents the replay of old "Integrity Response" messages. If the match is successful, the receiver saves the Sequence Number from the INTEGRITY object as the latest sequence number received with the key identifier included in the CHALLENGE.

If a response is not received within a given period, the challenge is repeated. When the integrity handshake succeeds, the receiver begins accepting normal RSVP signaling messages from that sender and ignores any other "Integrity Response" messages.

Implementations SHOULD enable the Integrity Handshake by default when RSVP Cryptographic Authentication is in use. In some special-case environments it might not be required. One use of RSVP Authentication might be between peering domain routers that are processing a steady stream of RSVP messages due to aggregation effects. When a router restarts after a crash, valid RSVP messages from peering senders probably will arrive shortly. If replay messages are injected into the stream of valid RSVP messages, there might be only a small window of opportunity for a replay attack before a valid message is processed. This valid message will set the largest sequence number seen to a value greater than any number before the crash, preventing any further replays. This attack can be mitigated if the router has stable storage for the RSVP sequence number state.

If an implementation does not enable the Integrity Handshake, this creates a broad exposure to replay attacks, especially if there is a long period of silence from a given sender following a restart of a receiver.

Hence, it SHOULD be an administrative decision by the network operator whether or not the receiver performs an Integrity Handshake with senders that are willing to respond to its "Integrity Challenge" messages, and whether it accepts any messages from senders that refuse to do so. These operational decisions ought to be based on risk assessments [NIST-RMF] for the particular network environment.

Each RSVP session MUST be protected either by the Integrity Handshake or by sequence numbers recorded in stable storage.

5. Key Management

Different operators have different approaches and methods for network device (e.g., switch, router) configuration management. Whichever configuration management method an operator uses for network device configuration also can be used to configure RSVP Authentication. It is beyond the scope of this document to mandate that an operator use a particular method for network device configuration management.

As of the publication date for this document, it appears very unlikely that the IETF will define a standard key management protocol for use with RSVP. However, if the IETF does so, then it would be strongly desirable to use that key management protocol to distribute RSVP Security Associations (including keys) among the communicating RSVP implementations. Such a protocol could improve scalability and significantly reduce the human administrative burden. The Key Identifier can be used as a hook between RSVP and such a future protocol.

Key management protocols have a long history of subtle flaws that often are discovered long after the protocol was first described in public [DS-1981]. To avoid changing all RSVP implementations if such a flaw is discovered, integrated key management protocol techniques were deliberately omitted from this specification.

5.1. RSVP Security Association

An RSVP Security Association consists of the following parameters:

* RSVP Key Identifier (48 bits)

This unsigned 48-bit item is the Key Identifier used on the wire in the RSVP INTEGRITY object.

* RSVP Message Processing Mode

This value indicates whether this RSVP Security Association is using per-interface message processing rules or is using per-neighbor message processing rules.

* RSVP Sending Interface

This is the implementation-specific name for the sending interface associated with this RSVP Security Association. This only exists (a) when the per-interface message processing rules are in use for this RSVP Security Association and (b) only on the sending RSVP node.

* RSVP Receiving Interface

This is the implementation-specific name for the receiving interface associated with this RSVP Security Association. This only exists (a) when the per-interface message processing rules are in use for this RSVP Security Association and (b) on the receiving RSVP node.

* RSVP Sending IP Address

This is the IP address (IPv4 or IPv6) used by the sending node. This is only pertinent when the per-neighbor message processing rules are used for this RSVP Security Association.

* RSVP Receiving IP Address

This is the IP address (IPv4 or IPv6) used by the receiving node. This is only pertinent when the per-neighbor message processing rules are used for this RSVP Security Association.

* RSVP Cryptographic Transform

This item specifies the combination of the cryptographic algorithm (e.g., MD5, SHA-1, SHA-256) to be used, the cryptographic authentication mode (e.g., GMAC, HMAC, KMAC) to be used, and the length in bits of the Authentication Key. RSVP Cryptographic Transforms are defined in an IANA Registry (defined below) and corresponding transform-specific RFCs.

* RSVP Authentication Key

This item specifies the cryptographic authentication key to be used. Its size varies depending on which Cryptographic Transform is in use. For any specific RSVP Cryptographic Transform, the key size will be fixed. RSVP Authentication Keys need to be cryptographically random.[RFC4086][NIST-ENTROPY]

* RSVP Security Association Start Time

This item, referred to as KeyStartValid in [RFC2747], specifies the calendar date (e.g., 01 June 1970) and 24-hour clock time (e.g., 18:05) when this RSVP Security Association begins being valid for operational use. This value MUST NOT be later than the RSVP Security Association End value.

* RSVP Security Association End Time

This item, referred to as KeyEndValid in [RFC2747], specifies the calendar date (e.g., 01 June 1970) and 24-hour clock time (e.g., 18:05) when this RSVP Security Association stops being valid for operational use. This value MUST NOT be earlier than RSVP Security Association Start value.

RSVP Cryptographic Authentication has always implicitly required that all communicating RSVP-capable devices have at least loosely synchronized clocks. In some cases, hardware clocks inside a network device might be sufficient. However, many network deployments use the Network Time Protocol (NTP), Precision Time Protocol (PTP), or another method to keep such clocks sufficiently synchronized. When possible, RSVP deployments SHOULD also deploy a distributed time synchronization protocol and SHOULD enable cryptographic authentication for that time synchronization protocol.

Certain key generation mechanisms, such as Kerberos or some public key schemes, might directly produce ephemeral keys for use with RSVP. In that case, the lifetime of the key MAY be defined as part of that key generation process.

In normal operation, an RSVP Security Association is never used outside its lifetime, but see Section 5.3 for a degenerative special case.

5.1.1. Additional State

Implementations will require additional state associated with, but not part of the Security Association. This information is not part of a Security Association’s configuration.

* Initial RSVP Authentication Sequence Number

For an RSVP Security Association which has not yet been used, this MUST be initialized to an unpredictable (i.e., cryptographically random) value.[RFC4086][NIST-ENTROPY] Both sender and receiver either MUST be configured with this initial sequence number or MUST learn it via the Integrity Handshake, so that the RSVP Authentication sequence number windowing scheme can work properly.

After the RSVP Security Association has been used by the sender, the sender uses the Latest Sent RSVP Authentication Sequence Number instead. After the RSVP Security Association has been used by the receiver, the receiver uses the List of Received RSVP Authentication Sequence Numbers instead.

* Latest Sent RSVP Authentication Sequence Number

This exists only on the RSVP Authentication sending node. It contains the most recently transmitted Sequence Number within the RSVP Integrity Object.

* List of Received RSVP Authentication Sequence Numbers

This list exists only on the RSVP Authentication receiving node. It contains an ordered list of the most recently seen sequence numbers. The list MUST support at least 5 sequence numbers and MAY support more than 5. This is used as part of the replay mitigation mechanism. It is a list rather than a single number because IP packets might be reordered in transit from a sender to a receiver.

5.2. Key Management Procedures

To maintain security, it is advisable to change the RSVP Security Association regularly. Operational considerations mean it needs to be possible to switch the RSVP Security Association smoothly (i.e., without loss of RSVP state or denial of its reservation service), and also without requiring people to change all the keys simultaneously.

Supporting smooth key rollover in an RSVP implementation is essential. Therefore, RSVP implementations MUST support the storage and use of at least 2 active RSVP Security Associations concurrently (a) for each RSVP-enabled interface when using the per-interface message processing rules and (b) for each RSVP peer when using per-peer message processing rules. To best support resilient network operations, the number of concurrent RSVP Security Associations SHOULD NOT merely be two (2), but instead SHOULD be a much larger number.

Since Security Associations are shared between an RSVP sender and one or more RSVP receivers, there is a region of uncertainty around the time of key switch-over during which some systems may still be using the old key and others might have switched to the new RSVP Security Association. The size of this uncertainty region relates directly to the clock synchrony of the systems. Administrators ought to configure the overlap between the expiration time of the older RSVP Security Association and the validity of its replacement RSVP Security Association to be at least twice the size of this uncertainty interval. In many deployments, five (5) minutes of RSVP Security Association overlap will suffice. This will allow the sender to make the RSVP Security Association switch-over at the midpoint of this interval and be confident that all receivers are now accepting the new Security Association. For the duration of the overlap in RSVP Security Association lifetimes, a receiver must be prepared to authenticate messages using either Security Association. The combination of the sender's IP address and the Key Identifier will inform the receiver of which Security Association to use for validation.

During rollover of the RSVP Security Association, it will be necessary for each receiver to handshake with the sender using the new RSVP Security Association. As stated above, an RSVP receiver has the choice of initiating a handshake during the switchover or postponing the handshake until the receipt of a message using that key.

5.3. Key Management Requirements

Requirements for an implementation are as follows:

* It is strongly desirable that a hypothetical security breach in one Internet protocol does not automatically compromise other Internet protocols. The Authentication Key of this specification SHOULD NOT (a) be stored in an insecure manner or (b) transmitted either in clear-text or using protocols, algorithms, or methods that have known flaws.

* An implementation MUST support the storage and use of more than one Security Association at the same time for the same interface (when per-interface processing rules apply) or for the same RSVP peer (when per-peer processing rules apply). It is impossible to support smooth key rollover if an implementation does not support at least 2 concurrent RSVP Security Associations of each type.

* An implementation MUST associate a specific lifetime with each RSVP Security Association and the corresponding RSVP Key Identifier.

* An implementation MUST support manual key distribution (e.g., the privileged user manually typing in all the RSVP Security Association parameters on the console). A manually entered RSVP Security Association lifetime MAY be used forever, although this is neither recommended nor best practice.

* Keys that are out of date MAY be automatically deleted by the implementation ONLY IF a replacement RSVP Security Association is already configured and active.

* Manual deletion of active keys (e.g., from an operator console) also MUST be supported.

* RSVP Security Association storage MUST persist across a system restart, warm or cold, to ease operational usage -- EXCEPT that the RSVP Sequence Number information need not be persistent across a system restart.

5.4. Pathological Case

It is possible, although strongly undesirable, that all applicable RSVP Security Associations have expired. If this happens, it is unacceptable to revert to an unauthenticated condition, and a disruption to current reservations could cause a broader network fault.

It is strongly recommended that network operators prevent this situation from occuring. Rekeying prior to sequence number rollover is an example of a way for network operators to avoid this situation.

Therefore, in that event, to keep the network operational the system SHOULD send a "last RSVP Security Association expiration" notification to the network manager (e.g., via SYSLOG or SNMP) and also SHOULD treat the RSVP Security Association as having an infinite lifetime until either (a) the RSVP Security Association’s lifetime is extended, (b) the RSVP Security Association is deleted by network management, or (c) a new RSVP Security Association is configured.

5.5. Kerberos

Use of Kerberos with RSVP Authentication is outside the scope of this document. Such use might be specified in the future in some other RFC.

6. Security Considerations

This entire memo describes and specifies an algorithm-independent authentication mechanism for RSVP that is believed to be secure against passive attacks and against most active attacks provided the selected cryptographic transform is secure, the RSVP Security Association is known only to appropriate devices, and the devices' RSVP implementations have been appropriately configured.

The quality of the security provided by this mechanism depends on the strength of the implemented authentication algorithms, the strength of the key being used, and the correct implementation of the security mechanism in all communicating RSVP implementations. This mechanism also depends on the RSVP Authentication Keys being kept confidential by all parties. If any of these assumptions are incorrect or operational procedures are insufficiently secure, then no real security will be provided to the users of this mechanism.

While the handshake "Integrity Response" message is integrity-checked, the handshake "Integrity Challenge" message is not. This was done intentionally to avoid the case when both peering routers do not have a starting sequence number for each other's key. Without this, both routers will each keep sending handshake "Integrity Challenge" messages that will be dropped by the other end. Moreover, requiring only the response to be integrity-checked eliminates a dependency on a security association in the opposite direction.

However, this allows a potential intruder to generate fake handshaking challenges with a certain challenge cookie. It could then save the response and attempt to play it against a receiver in recovery. If it were lucky enough to have guessed the challenge cookie used by the receiver at recovery time, then it could use the saved response. This response would be accepted, since it is properly signed, and would have a smaller sequence number for the sender because it was an old message. This opens the receiver up to replays. Still, this seems difficult to exploit. It requires not only guessing the challenge cookie (which is based on a locally known secret, possibly including a timestamp) in advance, but also being able to masquerade as the receiver to generate a handshake "Integrity Challenge" with the proper IP address without being caught.

Confidentiality and protection against traffic analysis are not provided by this mechanism. Mechanisms such as bulk link encryption (e.g., IEEE 802.1 MAC Security [IEEE-802.1AE-2018] for an Ethernet link) can be used to provide hop-by-hop confidentiality and some mitigation against traffic analysis. [NSA-MSCCP]

7. IANA Considerations

IANA is requested to create a new registry named "RSVP Cryptographic Transforms" within the existing "Resource Reservation Protocol (RSVP) Parameters" registry group.

It is helpful for implementers of this specification to know the current set of defined cryptographic transforms, the corresponding RFC(s) for each cryptographic transform, and the Implementation Status for each cryptographic transform.

Each registry entry will need to contain, the Name of the specific Cryptographic Transform (e.g., HMAC-MD5), the RFC(s) which specify that Method (e.g., RFC-2747), and the current Implementation Status of that Method. The Name of the Method is limited to printable uppercase US-ASCII letters, printable US-ASCII numbers, and the character "-". The "Implementation Status" field of any method MUST be one of the following values (MUST NOT, SHOULD NOT, MAY, SHOULD, or MUST) which are to be interpreted as per [RFC2119].

The RSVP Cryptographic Transforms registry can be updated by the IETF Review procedure (which procedure also allows updating via an IETF Standards Action). This means a new IETF Stream RFC will be required either to define a new RSVP cryptographic transform, to update the Implementation Status of one or more existing RSVP cryptographic transforms, or to do both.

There is one initial value in the new registry:

Name	Reference(s)	Implementation Status
HMAC-MD5	RFC 2747	SHOULD

8. Acknowledgments

This document’s predecessor, [RFC2747], was authored by Fred Baker, Bob Lindell, and Mohit Talwar. That was derived directly from similar work done for OSPF version 2 and RIP Version 2, jointly by Ran Atkinson and Fred Baker. Significant editing of the text in [RFC2747] was done by Bob Braden, resulting in increased clarity. Significant comments on [RFC2747] were submitted by Steve Bellovin. Matt Crawford and Dan Harkins also helped revise draft versions of [RFC2747].

In April 2001, [RFC3097] updated the RSVP message type value used for the RSVP Integrity Object to resolve an issue caused by a conflicting assignment.

Appendix A: Changes since RFC 2747

This document has made the following substantive changes since [RFC2747]:

* This specification is algorithm-independent and cryptographic-mode independent. So adding support for a new cryptographic algorithm or cryptographic mode will not require changes to this protocol specification. Those algorithm-dependent and mode-dependent specifications will be in separate RFCs, which can be standardized, recommended, and/or deprecated over time without changes to this RFC or protocol specification.

* The Authentication Data field of the INTEGRITY object now supports an increased length so that other algorithms can be supported. The reserved field has been repurposed to indicate any increased length of the Authentication Data field. This allows the authentication mechanism to support other algorithms and modes robustly.

* Discussions of Security Associations have been made RSVP-specific and moved to Section 5.1.

* Peer-specific Security Associations are explicitly supported.

* Implementation of the Integrity Handshake is now required.

* The discussion of Key Management has been updated.

* The discussion of Kerberos has been greatly reduced.

References

Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <https://www.rfc-editor.org/info/rfc2205>.

[RFC3097] Braden, R. and L. Zhang, "RSVP Cryptographic Authentication -- Updated Message Type Value", RFC 3097, DOI 10.17487/RFC3097, April 2001, <https://www.rfc-editor.org/info/rfc3097>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

Informative References

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC2408] Maughan, D., Schertler, M., Schneider, M., and J. Turner, "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, DOI 10.17487/RFC2408, November 1998, <https://www.rfc-editor.org/info/rfc2408>.

[RFC2747] Baker, F., Lindell, B., and M. Talwar, "RSVP Cryptographic Authentication", RFC 2747, DOI 10.17487/RFC2747, January 2000, <https://www.rfc-editor.org/info/rfc2747>.

[RFC2752] Yadav, S., Yavatkar, R., Pabbati, R., Ford, P., Moore, T., and S. Herzog, "Identity Representation for RSVP", RFC 2752, DOI 10.17487/RFC2752, January 2000, <https://www.rfc-editor.org/info/rfc2752>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC6151] Turner, S. and L. Chen, "Updated Security Considerations for the MD5 Message-Digest and the HMAC-MD5 Algorithms", RFC 6151, DOI 10.17487/RFC6151, March 2011, <https://www.rfc-editor.org/info/rfc6151>.

[RFC8937] Cremers, C., Garratt, L., Smyshlyaev, S., Sullivan, N., and C. Wood, "Randomness Improvements for Security Protocols", RFC 8937, DOI 10.17487/RFC8937, October 2020, <https://www.rfc-editor.org/info/rfc8937>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

[DS-1981] Denning, D. and G. Sacco, "Timestamps in Key Distribution Protocols", August 1981. Communications of the ACM, Vol. 24, No. 8

[G.8265.1] ITU-T, "Precision Time Protocol Telecom Profile for Frequency Synchronization", ITU-T Recommendation G.8265.1, July 2014.

[IEEE-802.1AE-2018]
Institute of Electrical and Electronics Engineers (IEEE), "IEEE Standard for Local and Metropolitan Area Networks - Media Access Control (MAC) Security", December 2018, <https://standards.ieee.org/IEEE/802.1AE/7154/>. IEEE Standard 802.1AE

[IEEE-1588-2019]
Institute of Electrical and Electronics Engineers (IEEE), "IEEE Standard for Precision Clock Synchronization Protocol for Networked Measurement and Control Systems (PTP v2.1)", June 2020, <https://ieeexplore.ieee.org/document/9120376>. IEEE Standard 1588

[NIST-ENTROPY]
Turan, M., "Recommendation for the Entropy Sources Used for Random Bit Generation", January 2018, <https://csrc.nist.gov/pubs/sp/800/90/b/final>. Special Publication 800-90B

[NIST-GMAC]
Dworkin, M., "Recommendation for Block Cipher Modes of Operation: Galois Counter Mode (GCM) and GMAC", November 2007, <http://csrc.nist.gov/Projects/message-authentication-codes>. Special Publication 800-38D

[NIST-HMAC]
(US) National Institute of Standards and Technology, Gaithersburg, MD, USA, "Keyed-Hash Message Authentication Code (HMAC)", July 2008, <http://csrc.nist.gov/Projects/message-authentication-codes>. (US) Federal Information Processing Standard 198-1 (FIPS-198-1)

[NIST-KMAC]
Kelsey, J., Chang, S.-J., and R. Perlner, "SHA-3 Derived Functions - cSHAKE, KMAC, TupleHash, and ParallelHash", December 2016, <http://csrc.nist.gov/Projects/message-authentication-codes>. Special Publication 800-185

[NIST-RMF] Joint Task Force, "Risk Management Framework for Information Systems and Organizations", December 2018, <http://csrc.nist.gov/Projects/risk-management>. Special Publication 800-37, Revision 2

[NSA-MSCCP]
(US) National Security Agency, Ft. Meade, MD, USA, "Multi-Site Connectivity Capability Package", June 2018, <https://www.nsa.gov/Resources/Commercial-Solutions-for-Classified-Program/Capability-Packages/>. Version 1.1

[WAGNER] Wagner, R., "Comments on Decision Proposal to convert FIPS-198-1 to a NIST Special Publication", October 2022, <https://csrc.nist.gov/csrc/media/Projects/crypto-publication-review-project/documents/decision-proposal-comments/fips198-1-decision-proposal-comments-2022.pdf>.

Authors' Addresses

Ran Atkinson
Consultant
United States of America
Email: rja.lists@gmail.com

Tony Li
Juniper Networks
United States of America
Email: tony.li@tony.li