End-to-End Transport of UNI Client Authentication, Integrity, and Data Plane Security Support Information

IA # OIF-E2E-SEC-01.0

June 4, 2012

Implementation Agreement created and approved
by the Optical Internetworking Forum
www.oiforum.com

The OIF is an international non-profit organization with over 90 member companies, including the world's leading carriers and vendors. Being an industry group uniting representatives of the data and optical worlds, OIF's purpose is to accelerate the deployment of interoperable, cost-effective and robust optical internetworks and their associated technologies. Optical internetworks are data networks composed of routers and data switches interconnected by optical networking elements.

With the goal of promoting worldwide compatibility of optical internetworking products, the OIF actively supports and extends the work of national and international standards bodies. Working relationships or formal liaisons have been established with IEEE 802.1, IEEE 802.3ba, IETF, IP-MPLS Forum, IPv6 Forum, ITU-T SG13, ITU-T SG15, MEF, ATIS-OPTXS, ATIS-TMOC, TMF and the XFP MSA Group.

For additional information contact:
The Optical Internetworking Forum, 48377 Fremont Blvd.,
Suite 117, Fremont, CA 94538
+1 510 492 4040
info@oiforum.com

www.oiforum.com

End-to-End Transport of UNI Client Authentication, Integrity, and Data Plane Security Support Information

ABSTRACT: This Implementation Agreement defines an optional extension to the OIF UNI. This extension consists of a UNI client service that supports authenticating and transparently transporting a UNI client's set of UNI message objects and additional security information across the entire signaling network between two UNI-C's. Authentication is performed with an end-to-end client digital signature mechanism that provides authentication, integrity, and support for non-repudiation of end-to-end UNI-to-UNI communications. It describes what data items to protect, how to apply this protection, policy specification and enforcement, security credentials, error and restart handing, performance, recovery, and security considerations. Using this extension depends on availability of an end-to-end signaling network for carrying security objects. In the cases where this violates carriers' policies, UNI clients cannot assume it is available. As a result, this Implementation Agreement applies only to those cases where mutual agreement exists between service providers and end clients.

Table of Contents

1. Introduction 1
1.1 Problem Statement 1
1.2 Scope 2
1.3 Background on End-to-End Protocol Security 3
1.4 OIF UNI End-to-End Security 4
1.5 Relationship to Other Standards Bodies 5
1.6 Acknowledgements 5
1.7 How to Use this Implementation Agreement 5
1.8 Document Organization 6
2. Terminology and Acronyms 6
2.1 Keywords 6
2.2 Terminology 6
2.3 Acronyms 7
3. Objectives and Requirements 8
4. RSVP Signature Option 10
4.1 Background on Digital Signatures 11
4.2 Overview of Signatures for the OIF UNI 11
4.3 Timestamps and Replay Detection 13
4.4 Names and Certificates 13
4.5 Distributing Certificates 14
4.6 Mapping from Names to CALL_IDs 15
4.7 What to Sign: Mutable, Immutable, and Excluded Objects 16
5. Data Structures, Error Handling, Logging, and Processing Rules 20
5.1 Structure of the OIF_E2E_SECURITY Subobject 22
5.2 Error Codes and Error Logging 28
5.3 Processing Rules 34
5.4 Backward Compatibility 35
6. Policy Considerations at UNI Endpoints 36
6.1 Specifying, Enforcing, and Changing Policies without Disruption 36
6.2 Signaling Channel Failure, Restart, and Policy Updates 37
7. OIF Assigned Numbers 38
8. Performance 38
9. Security Considerations 39
10. Short Signatures (Informative) 40
11. Summary 41
12. References 41
12.1 Normative References 41
12.2 Informative References 42
Appendix A: Glossary 45
Appendix B: OIF Members When the Document Was Approved 46

List of Figures

Figure 1: The OIF's UNI and E-NNI Signaling Interfaces (from [E-NNI]) 4
Figure 2: Structure of a Signature Block 12

List of Tables

Table 1: Example of Message Size for Singing a Path Message 39

Document Contact Information

TECHNICAL EDITOR

Richard Graveman
Department of Defense
15 Park Avenue
Morristown, NJ 07960 USA
Phone: +1 973 984 8780
Email: rfg@acm.org

WORKING GROUP CHAIRS

Doug Zuckerman, Telcordia Technologies

Evelyne Roch**,** Ciena Corporation

Rémi Theillaud, Marben Products

End-to-End Transport of UNI Client Authentication, Integrity, and Data Plane Security Support Information

1. Introduction

This document defines an optional extension to the OIF UNI. It consists of a UNI client service that supports authenticating and transparently transporting a UNI client's set of UNI message objects and additional security information across the entire signaling network between two UNI-C's. To verify this information, a method to generate and verify digital signatures on data items in RSVP-TE signaling messages at a user-network interface (UNI-C) reference point is defined. The objective is to define a mechanism that ensures the integrity and authenticity of the end-to-end user-defined items in such messages. Using this extension depends on availability of an end-to-end signaling network for carrying security objects. In the cases where this violates carriers' policies, UNI clients cannot assume it is available. As a result, this Implementation Agreement applies only to those cases where mutual agreement exists between service providers and end clients.

This document also defines how supporting security information can be communicated transparently, end-to-end between the UNI clients. The information exchanged is client specific (i.e., unspecified).

This document also provides material helpful to implementers and users of end-to-end UNI authentication and integrity. It covers topics such as what data items to protect, how to apply protection (i.e., handling mutable [original] and immutable data items), security policy specification and enforcement, security credentials, error handing, performance, security considerations, and restart and recovery considerations.

Parts of this document contain background information on how end-to-end UNI security works, advice to implementers on what tools and utilities to provide with this security mechanism, and advice to users on setting up and configuring security policy, handling errors, and logging.

1.1 Problem Statement

Briefly, the way RSVP-TE (and most other signaling protocols) works is:

- The message payload consists of a main header that defines the type of message followed by a list of type-length-value objects (TLVs)
- Each party along the path may, according to its policy and function, modify or fill in information in certain objects as appropriate
- Processing rules exist for changing or adding information, but no end-to-end method is defined for detecting whether these rules were followed or not
- The final receiver does not know what was originally sent versus what was added, changed, or deleted later

OIF has defined extensions for securing the inter-domain interfaces, i.e., UNI and E-NNI. The OIF's profile of IPsec and the IETF's RSVP INTERGITY object protect messages but only across a single RSVP session, so they cannot be used across multiple sessions. The main problem created by these semantics for the OIF UNI is that two UNI-C [UNI2.0, UNI2.0-RSVP] entities setting up, modifying, or tearing down a call do not have any direct assurance about what the other party has requested or even any strong assurance about who the other party is. This is an independent topic from service operators' requirements to hide internal details about *how* the users' requests are handled.

A possible side benefit of signing parts of such messages is that logging *signed* and *timestamped* messages provides much stronger evidence of end-users' actions in case audit logs disagree on what has been requested.

This optional extension applies only to implementations based on RSVP-TE [UNI2.0-RSVP] at the UNI. Furthermore, the OIF's UNI signaling model based on ASON requirements and defined in [UNI2.0] does not depend on any particular signaling protocol within the network. Various signaling approaches may be used to connect the two, independent UNI sessions, one at the source and one at the destination. Only limited information including the contents of the Generalized_UNI object is communicated between the source and destination UNI-N's. For this optional extension to work, an additional RSVP object, subobject, and its sub-subobjects need to be delivered transparently, intact between source and destination UNI-N's. Because doing this involves additional complexity and overhead, and because it passes information generated outside the network through the SCN, support for this extension is optional and depends on service operators for each domain between UNI-N's.

1.2 Scope

This optional client service for the OIF UNI and the supporting digital signature mechanism are intended primarily to work only between UNI-C entities. In a soft permanent connection (SPC), one or both of the connection endpoints is under management system control rather than UNI client control. Support for this service in this case requires additional functionality at the UNI-N which is for future study. Generalizations to other interfaces are also for future study. For example, signing OIF E-NNI NOTIFY messages, or indeed GMPLS signaling messages over RSVP-TE in general, may be useful, but defining trust models, handling key management, and potentially working with multiple signatures all add complications.

This digital signature mechanism is independent of and somewhat complementary to existing, hop-by-hop security mechanisms. Many of the objects at the OIF UNI are meaningful only between the UNI-C and UNI-N reference points and have no end-to-end significance. This IA is not intended to cover these objects.

The confidentiality of the UNI data items signed by this mechanism is out of scope.

This is an optional digital signature mechanism defined only for the OIF UNI [UNI2.0-RSVP]. The OIF UNI uses the RSVP-TE protocol defined by the IETF and

extended by the OIF. As such, it may not work with other signaling protocols on the path between the UNI endpoints.

The support of this client service and the applicability of this mechanism may vary according to the relationship the users on the two ends have with each other, the configuration of the signaling communications networks (SCNs) between them, and the policies enforced on these SCNs. For example, this mechanism may be used:

i. entirely within a single user's private domain. In this case, the user has complete control of the endpoints and the network, and the user wants to protect the end-to-end signaling against a compromised or malfunctioning intermediate signaling point.

ii. between users who trust each other across a single carrier's semi-trusted network. In this case the users want to protect their end-to-end signaling against a compromised or malfunctioning intermediate signaling point or an impersonation attack against the entire signaling network. The users need support for this mechanism from their carrier, and their carrier may enforce a network operator's policy stating which users are allowed to use this mechanism and how they use it.

iii. between cooperating but untrusting users across one or more carriers. In this case, the users can establish trust through an external public key infrastructure. In contrast to case (ii), this mechanism may offer protection against additional points of attack. In the future, for multiple-carrier scenarios, all carriers in the path will need to allow this mechanism.

1.3 Background on End-to-End Protocol Security

The end-to-end security problem is not specific to signaling protocols. Another good example is link state routing protocols. For OSPFv2 (which is primarily used in intra-domain routing), see, for example, the discussion and solution in RFC 2154 [RFC2154]. End-to-end security for inter-domain routing has been addressed more recently in the IETF's SIDR Working Group.

Prior work on RSVP security has considered this and similar problems. Wu et al. [Wu99] stated the following in the abstract to their RSVP security paper:

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

Talwar and Nahrstedt [TN00] proposed using a combination of digital signatures for the immutable parts of an RSVP message and hop-by-hop message authentication codes for the mutable parts.

Behringer, Le Faucheur, and Weis [BFW11] have addressed a different problem: how to distribute shared keys among more than two parties for hop-by-hop integrity checks or confidentiality mechanisms. They cite the open problem of a subverted node:

> A subverted node is defined here as an untrusted node, for example because an intruder has gained control over it. Since RSVP authentication is hop-by-hop and not end-to-end, a subverted node in the path breaks the chain of trust. …

These references, with respect to both RSVP and protocols with similar characteristics, illustrate that client end-to-end assurance and solutions using digital signatures have been identified repeatedly as long-standing security challenges.

1.4 OIF UNI End-to-End Security

Figure 1 illustrates the OIF's UNI and E-NNI reference points in an overall control plane architecture. Each UNI-C entity (labeled Client) communicates with its UNI-N (its directly connected NE). Even though pairs of UNI-C's manage calls and connections between them, they do not use any signaling protocol to communicate directly with each other. This, in itself, explains why the OIF's existing RSVP security mechanisms provide no security assurances between pairs of UNI-C's. When two UNI-C's use RSVP-TE to manage calls and connections, certain RSVP message types sent by one UNI-C cause the same message types to be delivered to the other UNI-C, and certain RSVP objects or parts of objects within these messages are defined to have end-to-end significance. Providing client authentication, integrity, and freshness guarantees for these message types and data items is what is meant by "End-to-End Transport of UNI Client Authentication, Integrity, and Data Plane Security Support Information."

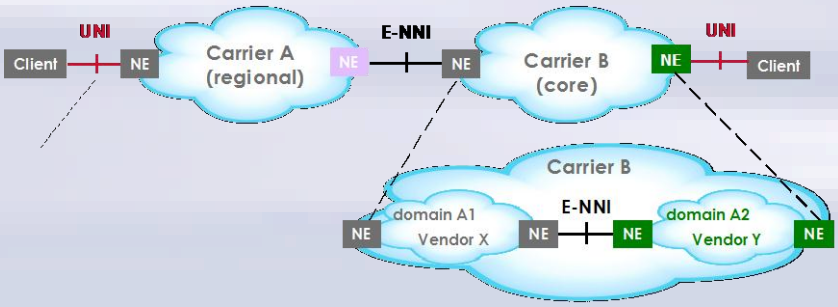


Figure 1: The OIF's UNI and E-NNI Signaling Interfaces (from [E-NNI]).

The IETF has defined the RSVP INTEGRITY object [RFC2747], which provides a secure checksum for messages between RSVP entities. The OIF has defined a security encapsulation mechanism that can protect signaling and routing messages between two UNI or between two E-NNI reference points [SecExt]. These security protocols protect what is sent between a UNI-C and a UNI-N or between two E-NNIs but give UNI-C entities no end-to-end way to verify which UNI-C is actually at the other end and what was requested or confirmed at the other end. This mechanism provides such end-to-end UNI

assurances in an in-line, immediate way, before committing to allocate networking resources.

In addition, when two UNI-C's set up calls and connections, they may need to exchange security setup information for their transport network elements' data plane connections. The details of how they secure data plane connections depend on the data plane technology and are beyond the scope of OIF control plane protocols. This optional mechanism also defines a sub-subobject to provide a single, authenticated way to exchange security setup information for such data plane connections (see Section 5.1.9). Whereas other mechanisms have been defined for user channels over specific transport technologies, this IA defines a method based on the OIF control plane. Because this sub-subobject needs to be delivered transparently and intact between UNI-C's, support for this sub-subobject is optional and depends on agreement with the network service operator. Note that this data plane security, which is applied to user's data (e.g., the payload in an Ethernet or SONET/SDH frame), must not interfere with the need to access or modify transport layer network information such as headers, trailers, or overhead along the data plane connection.

1.5 Relationship to Other Standards Bodies

One goal of the design in this IA is to simplify implementation by reusing algorithms and data structures already defined for the OIF's control plane. To that end, this IA reuses the OIF's UNI 2.0 signaling, the IETF's NTP [RFC5905], and, for security, the following work from the IETF:

- X.509 certificates as used in IKEv2 [RFC5996]
- Hash and URL of X.509 certificates as used in IKEv2
- SHA-1 and DSS as used in IKEv2
- OCSP as used in IKEv2
- Timestamps as used in syslog [RFC5424]

The main cryptographic methods (the SHA-1 hash function and the Digital Signature Standard) were defined by NIST. The certificate format, X.509 [X.509], was defined by the ITU-T.

1.6 Acknowledgements

Fred Gruman (Fujitsu), Jim Jones (Alcatel-Lucent), Monica Lazer (AT&T), Scott McNown (DoD), Thierry Marcot (France Telecom), George Newsome (Ciena), Lyndon Ong (Ciena), Evelyne Roch (Ciena), Jonathan Sadler (Tellabs), Stephen Shew (Ciena), Vishnu Shukla (Verizon), Chuck Sannipoli (IP Infusion), and Rémi Theillaud (Marben Products) provided helpful comments that led to improvements in this work.

1.7 How to Use this Implementation Agreement

This document defines an optional OIF UNI extension to support an end-to-end client service for authenticating signaling operations and transparently transporting user-defined supporting security information between UNI-C reference points.

It uses a private RSVP object (OIF_VENDOR_PRIVATE_ EXTENSION_TYPE_3) with the OIF's Enterprise Number as defined in [PrivExt], and it defines a new subobject (OIF_E2E_SECURITY) of this object. Implementing this service and subobject is OPTIONAL, but, as described in [PrivExt], intermediate protocol controllers that do not recognize this object and subobject pass them on unchanged, as long as all policy-enforcing protocol controllers, according to local operators' policies, allow this object and subobject to pass. Support for this extension beyond the UNI-N (towards the network) depends on the service operator's policy and agreements with the client. As required in Section 5, operators must have the capability to configure support for this subobject, and the default must be set to "off" or unsupported.

1.8 Document Organization

This document is organized as follows:

- Section 2 defines the terminology and acronyms used.
- Section 3 discusses requirements and objectives.
- Section 4 contains background on how the signature mechanism works and how to use it. It covers digital signatures, names, timestamps, certificates, distributing certificates, and selecting what to sign and how to sign it.
- Section 5 defines the signature mechanism. Section 5.1 covers data structures; Section 5.2 lists error codes and describes error handling and logging; Section 5.3 lists processing rules.
- Section 6 provides information on end-user policy, restart, and recovery.
- Section 7 lists OIF codepoints used by this mechanism.
- Section 8 discusses performance aspects: processing and communications.
- Section 9 addresses security considerations.
- Section 10 contains informative material on short digital signatures.
- Section 11 contains a summary, and Section 12 contains normative and informative references.

2. Terminology and Acronyms

2.1 Keywords

When written in ALL CAPITALS, the key words "MUST", "MUST NOT," "REQUIRED," "SHALL," "SHALL NOT," "SHOULD," "SHOULD NOT," "RECOMMENDED," "MAY," and "OPTIONAL" in this document are to be interpreted as described in IETF RFC 2119 [RFC2119].

2.2 Terminology

In this implementation agreement, the following definition applies:

Signature Block: The bytes that are signed. See Figure 1.

2.3 Acronyms

The following acronyms or abbreviations are used in this implementation agreement:

ASON	Automatically Switched Optical Network
CN	Common Name
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
E-NNI	External Network-Network Interface
GMPLS	Generalized Multiprotocol Label Switching
I-NNI	Internal Network-Network Interface
IA	Implementation Agreement
IANA	Internet Assigned Numbers Authority
IETF	Internet Engineering Task Force
IPsec	Internet Protocol Security
IKEv2	Internet Key Exchange version 2
ITU-T	International Telecommunication Union—Telecommunication Standardization Sector
IV	Initialization Vector
LSP	Label Switched Path
NAT	Network Address Translation
NNI	Network-Network Interface
NTP	Network Time Protocol
OCSP	On-line Certificate Status Protocol
OIF	Optical Internetworking Forum
OSPF	Open Shortest Path First
OSPFv2	Open Shortest Path First version 2
PEM	Privacy Enhanced Mail
PGP	Pretty Good Privacy
RFC	Request for Comments
RSVP	Resource Reservation Protocol
SD-ID	Structured Data Identifier
SHA	Secure Hash Algorithm
S/MIME	Secure Multipurpose Internet Mail Extension
SPC	Soft Permanent Connection
SSH	Secure Shell
TE	Traffic Engineering

TLS Transport Layer Security
TLV Type, Length, Value
TNA Transport Network Assigned (Name)
UNI User-Network Interface
UNI-C User-Network Interface—Client
UNI-N User-Network Interface—Network
URL Uniform Resource Locator

3. Objectives and Requirements

Because only certain items in UNI signaling messages have end-to-end significance, providing end-to-end authentication requires a client first to identify or to replicate, second to sign relevant parts of the original message, and third to pass this signature end to end. In high-volume, low-revenue-per-connection switching applications (i.e., voice), where the protocol designers and implementers are counting bits per message and calls per second, such an idea entails huge overhead both in message size and processing requirements. Optical switching, on the other hand, may be a lower-volume, higher-value-per-connection service, and the tradeoffs may be different. With today's processing speeds, allowing UNI clients to sign and verify signaling messages (or parts of messages) to obtain end-to-end assurance before allocating resources may be a viable security enhancement for UNI clients.

This IA does not replace the need to implement security mechanisms for control plane exchanges over UNI and E-NNI interfaces [SecExt]. It extends security coverage from a single interface to end-to-end client interactions with respect to the following security requirements [CarrierReq]:

- *R269: All Control Plane protocols shall include optional and interoperable security mechanisms (a) to authenticate entities exchanging information across an interface; (b) to guarantee the integrity of the information exchanged across an interface and to detect replay attacks; (c) to protect the confidentiality of information that communicating entities may be required to keep secret from other parties.*

  The end-to-end authentication and integrity mechanism defined in this IA works across more than one interface. This end-to-end authentication and integrity mechanism may help end clients to identify forged or improperly modified signaling messages that occur during their exchanges.

- *R270: These security mechanisms shall protect against passive eavesdropping and active attacks against the optical network as well as unintentionally malfunctioning control entities (for example, due to software or configuration errors).*

  End-to-end authentication and integrity for control plane messages can identify certain unexpected end-to-end signaling behaviors and detect active attacks in

certain configurations not protected by other security mechanisms. (It does not protect against passive eavesdropping.)

- *R271: These security mechanisms shall be designed to prevent or limit the effect of denial of service attacks.*

  End-to-end authentication and integrity mechanisms can help identify and stop denial-of-service attacks against end clients.

- *R272: These security mechanisms shall be designed so they can be extended to incorporate or accommodate the particular or proprietary needs of individual users and be kept up to date with advances in security technology.*

  This IA, as explained above for R269 and R270, supplements security across a single interface and keeps the OIF's security work in step with new end-to-end security work on other protocols (e.g., BGP4). It may also be of particular interest to end-users with high-assurance security requirements.

- *R273: Tools and methods shall be included with these security mechanisms to specify and configure them based on policy, operate them with minimal manual intervention, and audit their correct operation.*

  This IA is consistent with this requirement. It includes support for policy and logging.

- *R274: To reduce implementation cost, improve manageability, enhance interoperability, reduce the risk of errors, and provide compatibility with other protocols, these security mechanisms should be based on a minimal, well-understood, and widely used set of cryptographic primitives at the network or transport layer and a comprehensive key management system that can be used with all Control Plane protocols (e.g., signaling, routing, and discovery).*

  All of the cryptographic methods used in this IA are drawn from existing standards that are widely used and occur already in existing OIF security IAs.

- *R275: The security system shall provide a mechanism to specify and enforce a security policy that states where and when security services must be applied.*

  This IA is consistent with this requirement.

Although this mechanism is defined entirely at the UNI endpoints, it does extend the OIF UNI signaling model and add message overhead across the entire end-to-end signaling network. This overhead is examined in greater detail in Section 8.

Two constraints on the design are (1) transparency to and minimal impact on nodes not implementing this feature and (2) placement high enough in the protocol stack to survive end to end.

One goal is to add end-to-end signatures to messages in a way that is transparent to entities not knowing about signatures. This allows for partial deployment and ensures backward compatibility. A second goal is to make this mechanism optional and ensure that it has little

to no impact on entities not supporting it. A third goal is to avoid translating messages with signatures into messages without signatures or building tunnels or other communications channels to hold signatures of messages. A fourth goal is to avoid introducing new messages and to minimize message expansion. Protocol design principles dictate that the best place to authenticate a protocol message is directly in the message itself. All of the common examples—IPsec, TLS, SSH, and S/MIME—do this. It is, overall, more efficient and more reliable. There are no additional messages and fewer things can go wrong. In the case of end-to-end UNI authentication, two more reasons exist. In-line authentication allows all parties to detect attacks that may improperly reserve or allocate costly resources to be detected as quickly as possible, and carrying authentication information on other channels may be technology dependent and require multiple solutions.

Signatures have to work end to end, so they need to be applied above the network layer, where NAT and other effects may interfere with proper operation. Signatures need to be applied to particular parts of the payload, but these parts or other parts of the payload still may need to be modified by the signaling protocol. The data structures need to accommodate these properties efficiently.

Security requirements for this mechanism include:

- Security policy enforcement at endpoints
- Message origin authentication
- End-to-end integrity for certain objects in a message
- Integrity for certain objects as they existed in the initial message while allowing for legitimate changes to these objects
- Detection of insertion of objects not in the original message
- Replay detection
- Support for non-repudiation of origin

Other requirements include:

- Transparent end-to-end operation
- Small or no impact on entities not implementing this feature
- Minimal message size expansion

4. RSVP Signature Option

This section provides background information needed to understand the working of this end-to-end UNI signature mechanism. Section 4.1 explains the operation of digital signatures, and Section 4.2 covers how signatures are used with RSVP-TE. Section 4.3explains how timestamps are used for replay detection. Section 4.4 defines how names and certificates work, and Section 4.5 covers how certificates can be installed where they are needed. Section 4.6 explains how messages can be mapped back to names, even though the names are not present in most messages. Finally, Section 4.7 lists the data items that should be protected in each message and how they should be protected.

4.1 Background on Digital Signatures

A digital signature is a string of bits, which an originator may add to a message to allow anyone possessing the public verification key to check that the originator indeed generated the message and that it has not been altered.

Digital signatures work by establishing for each party two keys, a signing key and a verification key. The signing key must be kept secret by the signer. The verification key can be distributed to all parties, but it must be protected against forgery or substitution. That is, the verifier must be sure that it has not been tricked into using a verification key belonging to an imposter. Of course, to be a secure signature scheme, there must be no way for someone with just a verification key and samples of signed messages to calculate the corresponding signing key or to produce forged signatures for plausible messages. For an overview of digital signatures and diagrams illustrating how they work, see Section 3 of [FIPS186-3]; for a thorough description of the cryptographic theory of digital signatures, see [Katz].

The signatures described in this work follow the Digital Signature Algorithm (DSA) specified by NIST. They are defined in Section 4 of [FIPS186-3] with $L = 1024$ and $N = 160$, which results in signatures that are 320 bits (40 bytes) long. Signing and verifying operations must be careful to ensure that all of the values chosen and computed satisfy the randomness requirements and range checks in the specification.

To use these signatures, implementations have to generate a new value k for each signature. Every value of k must be unique, unpredictable, and secret. However, it is possible to pre-compute pairs of values k and k^{-1} to make the on-line signing process more efficient. Because of these requirements, implementations need to have a source of cryptographically secure pseudo-random numbers. For cryptographic applications, pseudo-random numbers need stronger unpredictability properties than merely satisfying certain statistical tests. For more information on this topic, see [NIST800-90], [RFC4086], [Koç09], [Gut98], or [KSF99].

Alternatively, given a signing key and a message to be signed, a way to generate such an unpredictable, secret, pseudo-random value in a deterministic, message-dependent way is described in [Pornin].

4.2 Overview of Signatures for the OIF UNI

Because UNI signaling messages are not delivered intact, end to end, it would not be useful to apply a digital signature to an entire UNI signaling message. Therefore, end-to-end authentication and integrity for the OIF UNI is defined by applying digital signatures to certain data items in signaling messages that have end-to-end significance. A new subobject is inserted into RSVP messages for end-to-end delivery between two UNI-C's. Because the OIF UNI model is not based on one UNI-C to UNI-C signaling session, end-to-end delivery of this subobject depends on signaling interworking and support by the network of multiple concatenated signaling sessions. This subobject may contain:

i. the RSVP signaling message type (Section 5.1.1)
ii. a list of copies of "original" objects in the message (Section 5.1.3); copies of these objects as they existed at the source UNI are carried in the signature subobject, so that the destination UNI can determine whether they changed and whether such changes are appropriate
iii. a list of pointers to "immutable" objects in the message (Section 5.1.2); these objects are signed but not duplicated, so the signature can be used to detect whether all of these objects are the same at the receiver as they were at the sender
iv. lists of objects that did not appear in the message at the origin (Sections 5.1.4 and 5.1.5)
v. items needed to support data plane security, for example, key agreement information, initialization vectors, or synchronization tokens (Section 5.1.9)
vi. a strictly increasing timestamp used to detect replays (Section 5.1.6)
vii. pointers to the certificate containing the signature verification key and other certificates as needed (Section 5.1.7)
viii. requests and responses for certificate revocation information (Section 5.1.9)
ix. a signature (Section 5.1.8)

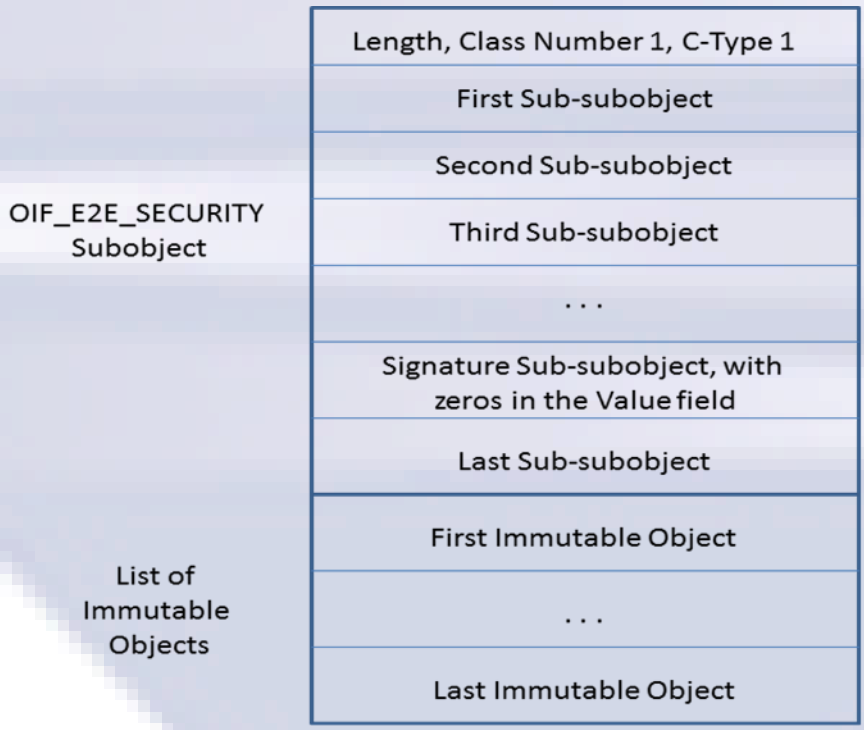


Figure 2: Structure of a Signature Block.

The only reason that the "immutable" construct exists is to save space in the message. Everything signed could be signed as original and replicated, but if it is known that an item

needs to be delivered as originally sent, it can be listed as immutable and not duplicated. It is important to note that the notion of "immutable" imposes no constraints on the network. There is never any implication that what the signature subobject asserts should alter the network behavior of the network.

Parts (i) through (viii) of this subobject together with the listed immutable objects make up what is signed. The actual data structure containing the bits that are signed is called a "signature block," as shown in Figure 2. The signature block, itself, is not transmitted intact. It is constructed by the sender and receiver at each end to generate and verify the signature, respectively.

4.3 Timestamps and Replay Detection

Replays of old messages are a serious threat to integrity. They are usually prevented by including counters, one-time values (called nonces), or timestamps in the authenticated portion of a message, or, in this case, in a signature block. With this end-to-end UNI mechanism, timestamps are used. They are written in ASCII text as specified for syslog [RFC5424] with a granularity of one microsecond, which is sufficient for any conceivable signaling application. These timestamps SHOULD always be used.

Senders MUST ensure that they use strictly increasing timestamps for each source-destination pair. Therefore, strict time synchronization is not needed, although keeping accurate time is useful for other purposes such as audit logging. Receivers merely need to keep track of the most recently received timestamp from each signer and check that each newer one is later. Note that the requirement to send increasing timestamps holds, even though the Network Time Protocol (NTP) or other mechanisms do not guarantee against adjusting clocks backwards.

4.4 Names and Certificates

Knowing who signed a message (or a signature block in this case) is an essential part of using digital signatures. Therefore, a certain amount of formality is needed to ensure that this part of the system cannot be spoofed.

First, names must be sufficiently unambiguous, so some well-defined namespace and syntactical rules such as email addresses, URIs, or DNS names are often chosen. The signatures defined here use the OIF UNI's TNA names (see [UNI2.0]). More exactly, a TNA name can have any of three formats with different lengths (32 bits, 128 bits, or 160 bits), so a TNA name is specified by the pair consisting of its <format, value>.

Second, certificates are used to make sure that verifiers map names to keys correctly. They allow a UNI client to authenticate its TNA name with its signature. A certificate contains (1) a UNI client's TNA name; (2) the UNI client's signature verification key; and (3) another digital signature certifying the binding between the two. The signature verifier relies on the party that signed the certificate, called a Certification Authority or CA, to have verified the relationship between TNA name and public key correctly.

Certificates have a formal structure specified by the ITU-T's Recommendation X.509, profiled by the IETF in RFC 5280, and used by the IETF in IKEv2 (RFC 5996). Certificates are written in Abstract Syntax Notation One (ASN.1).

Carriers may issue these certificates when they assign TNA names. On the other hand, in the absence of carrier-supported certificates that serve to assign TNA names, parties using this mechanism need to have their own way for certifying the TNA names they will use. They may use, for example, an external CA trusted by both or a CA internal to some organization to which they both belong.

If the CA signing a certificate is not known to the relying party, then it still may be possible to use a chain of certificates to establish trust. Many schemes based on X.509, therefore, arrange certificates in a hierarchy and widely distribute the certificate belonging to the top (called "root CA") of the hierarchy.

In addition to containing a signature from a trusted party, certificates must also be valid in other respects. They contain a serial number, the issuing CA's name, identifiers for the cryptographic algorithms used with the verification key and in the certificate's signature (which may be different), initiation and expiration dates, and possibly restrictions on how they may be used. CAs may also revoke certificates by publishing Certificate Revocation Lists.

The UNI client's TNA name is contained in the Subject Alternative Name extension of a certificate. The RECOMMENDED way to include TNA names in certificates is to ignore the CN and include the TNA sub-types plus names in one or more subjAltName extensions (see [RFC5280], Section 4.2.1.6). Thus, a single certificate may be valid for multiple TNA names. (Specifying TNA names with wildcards is for future study.)

Some connections may be initiated or terminated through management system control rather than a UNI client, i.e., soft permanent connections (SPCs). Support for transport of UNI client end-to-end authentication and security information for SPC connections requires additional functionality at the UNI-N and is for future study.Because the verifier relies on the issuer for the binding between TNA name and verification key, secure processes for issuing and revoking certificates may be required. These processes may require presenting appropriate credentials, performing authorization checks, demonstrating knowledge of the signing key, and taking delivery of the certificate in secure ways specified by a CA.

4.5 Distributing Certificates

One important design criterion for this end-to-end authentication mechanism is to minimize message expansion. This is why, for example, immutable objects are not replicated but merely listed. A signature is unavoidable, but certificates, which contain keys, names, signatures, and other items, are much longer than signatures. Therefore, two ways are provided to avoid having to encode actual certificates in signaling messages.

First, if there is no reason to believe that the receiver already has a certificate needed to verify a signature, then a URL telling the receiver where to find the certificate and a hash

(i.e., a cryptographically collision and preimage resistant checksum) of the certificate are signed and sent. The hash plus URL should be much shorter than the certificate.

Second, the protocol provides a way to point to a certificate that was previously made available to the other party. A certificate previously delivered with the hash-and-URL method should be remembered, so that it does not need to be sent, looked up, and verified again.

Also, certificates can be pre-installed, so that they never need to be communicated in signaling messages or by URL lookup. Implementers SHOULD provide tools to do this. Receivers keep an indexed array of certificates associated with a given signer, so the index number suffices to specify which certificate was used. This way, more than one certificate can be associated with each other signing party, so expiring and updating certificates cause no interruption in service.

4.6 Mapping from Names to CALL_IDs

To be able to verify these signatures, one has to know who (i.e., what TNA name) is on the other end of a signaling exchange. However, this information is not explicitly provided in every signaling message. To understand how to do this, the signaling messages need to be examined in more detail. UNI signaling provides three basic capabilities: (1) call and connection setup; (2) call and connection modification; and (3) connection release. (The release of the last remaining connection terminates a call.)

The RSVP-TE message types defined in the OIF UNI and used in messages with end-to-end significance are Path, Resv, ResvConf, PathErr, and PathTear. Message flows for call and connection setup, modification, and release need to be considered both when they successfully complete and when they fail or result in various error conditions. The following list shows the signaling action, RSVP-TE messages used to perform each signaling action, and directionality of the messages (where > means source to destination UNI-C and < means destination to source UNI-C):

- Call (or connection) setup: (Path, >), (Resv, <), and (ResvConf, >)
- Call (or connection) setup rejected by destination UNI-C: (Path, >), (PathErr , <)
- Call modification, adding a connection: (Path, >), (Resv, <), and (ResvConf, >)
- Call modification attempt, additional connection rejected by destination UNI-C: (Path, >), (PathErr, <)
- Successful connection modification, modifying service parameters: Path (P, >), Resv (R, <), (ResvConf , >), (Path, >), (PathErr, <), (Resv, <), (ResvConf, >)
- Unsuccessful connection modification, failure to increase or decrease bandwidth at destination UNI-C: (Path, >), (PathErr, <)
- Source UNI-C initiated connection release: (Path, >), (PathErr, <)
- Destination UNI-C initiated connection release: (Resv, <), (PathTear, >)

The Source and Destination TNA names are present only in the Path message. The initial exchange of Path and Resv messages establishes a CALL_ID. The CALL_ID is then present in every message except the ResvConf. Therefore, to be able to identify the TNA names on each side of a message, it suffices to be able to map CALL_IDs back to TNA names and to handle the ResvConf message.

A Local Connection Identifier is used to identify a connection uniquely at a UNI. With the OIF's RSVP-TE UNI signaling, the UNI_IPv4_SESSION object is included in all five messages with end-to-end significance and serves as a unique Local Connection Identifier that remains the same for the lifetime of the connection, even when connection modification occurs. Therefore, the following process can be used:

1. Source → Destination, Path: The Source fills in the Source and Destination TNA names and sets CALL_ID = 0. The Source signs and sends the message. It also remembers the TNA names and the UNI_IPv4_SESSION object it constructed, so that it can identify the reply.
2. The network fills in the CALL_ID and delivers a Path message with the two TNA names and signature to the Destination. The Destination verifies the signature and associates this CALL_ID with the Source and Destination TNA names for the rest of the call. Whenever the Destination requests a ResvConf message, it remembers how to associate the UNI_IPv4_SESSION object in the expected ResvConf message with the corresponding CALL_ID. Thus, the Destination can map every message with end-to-end significance to the proper pair of TNA names for the rest of the call.
3. Destination -> Source, Resv or PathErr: The Destination sends a signed Resv or PathErr with the CALL_ID. The Source receives a Resv or PathErr with CALL_ID, UNI_IPv4_SESSION, and signature. It uses the UNI_IPv4_SESSION object to check that the signature belongs to the correct TNA name, verifies the signature, and associates this CALL_ID with the TNA name it remembered. If the Source ever requests a ResvConf message, it uses the remembered UNI_IPv4_SESSION object, as described above for the Destination. Thus, the Source can map every message with end-to-end significance to the proper pair of TNA names for the rest of the call.

4.7 What to Sign: Mutable, Immutable, and Excluded Objects

This section enumerates the UNI data items that may be covered by end-to-end security. Table 7 of [UNI2.0-RSVP] summarizes the UNI message types and data items with end-to-end significance. Implementers should refer to this list as well as the information here for guidance on what to allow in signatures. Users MUST be allowed to configure a security policy at a UNI that states, for each other UNI, which items in which messages need to be signed with which keys. All of the signed items MAY be signed as original, but using the immutable option reduces message expansion. When using the immutable option, the signature verification will fail if the object is not delivered as signed. Therefore, if there is uncertainty about, for example, ordering of subobjects, canonicalization, or values of padding or reserved fields, signing as original may be necessary.

Five RSVP-TE message types are used in the OIF UNI but have no end-to-end significance, so they never occur with this signature mechanism: Hello, Ack, Srefresh, Bundle, and Notify.

The other five RSVP-TE message types do contain objects with end-to-end significance that may be signed:

1. Path
2. Resv
3. ResvConf
4. PathErr
5. PathTear

It is important always to include the RSVP message type in the signature to prevent attacks in which one signed message is substituted for another of a different type. To avoid extraneous message expansion, data items in the message with end-to-end significance should only be signed when their security is needed to accomplish a well-defined purpose. Certain objects, however, are needed in the signature to allow the destination to identify the source and to find the correct signature verification key. Finally, certain objects or parts of objects are not supposed to be delivered end to end, and extra original copies of these objects or subobjects, therefore, should not be included in what is signed.

The RSVP-TE objects that have end-to-end significance in these messages are listed in Table 7 of [UNI2.0-RSVP] and repeated here with short descriptions:

- ADMIN_STATUS in a Path or Resv message contains flags to indicate teardown of a connection.
- CALL_ID is filled in by the network for the first Path message of a call, and it is subsequently used as a call identifier in Path, Resv, PathErr and PathTear messages.
- The FLOWSPEC object for SONET/SDH, G.709, or ETHERNET returned in a Resv or ResvConf message contains the technology-dependent traffic parameters reserved for a connection.
- GENERALIZED_LABEL_REQUEST must be in a Path message. It asks for a label binding and includes an Encoding Type (e.g., SONET/SDH), Switching Type (e.g., TDM) and Generalized Payload Identifier (i.e., packet type, e.g., IPv4).
- GENERALIZED_ UNI_ATTRIBUTES must be in a Path message. It contains the SOURCE_TNA and DESTINATION_TNA. These are the identifiers that the two UNI-C's use to authenticate each other. However, they are not present in in other message types. All other items in this objects do not have end-to-end significance.
- IPv4_ERROR_SPEC in a ResvConf or PathErr message contains codes indicating specific errors.

- IF_ID ERROR_SPEC in a ResvConf or PathErr message contains codes indicating errors pertaining to specific interfaces.
- The SENDER_TSPEC object for SONET/SDH, G.709, or ETHERNET in a Path or PathErr message contains technology-dependent traffic parameters requested for a connection.
- STYLE in the Resv, ResvConf, and PathTear messages indicates whether or not connection modification is supported. It is always set to Shared Explicit (SE) or Fixed Filter (FF), respectively.
- SESSION_ATTRIBUTE is like STYLE, except it occurs in the Path message.

A Path message provides the Tspec to describe the traffic parameters for the desired connection, and a Resv message provides the Flowspec to describe the reservation the connection will use. These objects have different contents for the three supported technology groups, SONET/SDH, OTN, and Ethernet.

4.7.1 Path Message

In a Path message, the objects with end-to-end significance are:

1. ADMIN_STATUS

   This object should be signed. It may be signed as immutable.

2. CALL_ID

   For the first Path message of a call, this object is sent as zero and filled in before being delivered to the destination UNI-C. In this case it should be protected as original. In all other cases, it needs to be signed to let the Destination determine the who the Source is and may be signed as immuatable.

3. GENERALIZED_LABEL_REQUEST

   This object contains a LSP Encoding Type (i.e., Switching Type) and G-PID. It may be signed as immutable.

4. GENERALIZED_ UNI_ATTRIBUTES

   This object contains SOURCE _TNA, DESTINATION_TNA, DIVERSITY, and some number of occurrences of EGRESS_LABEL or SPC_LABEL and SERVICE_LEVEL. The Source UNI needs to sign this object as Original after removing all subobjects except the SOURCE _TNA and DESTINATION_TNA and reducing the length accordingly. The Desination UNI uses the SOURCE_TNA to pick a signature cerification key. It should check that the SOURCE _TNA in the outer message matches what was signed and that the DESTINATION_TNA matches its own identity.

5. SENDER_TSPEC for SONET/SDH, G.709, or ETHERNET

   This object should be signed. It may be signed as immutable.

6. SESSION_ATTRIBUTE

This object should be signed. It may be signed as immutable.

4.7.2 Resv Message

In the Resv message, the objects with end-to-end significance are:

1. ADMIN_STATUS

   This object contains flags to distinguish setup or modification from release of a connection. It should be signed. It may be signed as immutable.

2. CALL_ID

   This object needs to be signed and may be signed as immutable. It allows the Destination to determine who the Source is.

3. FLOWSPEC

   This object should be signed. It may be signed as immutable.

4. STYLE

   This object should be signed. It may be signed as immutable.

4.7.3 ResvConf Message

In the ResvConf message, the objects with end-to-end significance are:

1. FLOWSPEC

   This object should be signed. It may be signed as immutable.

2. IPv4_ERROR_SPEC

   This object should be signed as original.

3. IF_ID ERROR_SPEC

   This object should be signed as original.

4. STYLE

   This object should be signed. It may be signed as immutable.

4.7.4 PathErr Message

In the PathErr message, the objects with end-to-end significance are:

1. CALL_ID

   This object needs to be signed to allow the Destination to determine who the Source is. It may be signed as immutable.

2. IPv4_ERROR_SPEC

   This object should be signed as original.

3. IF_ID ERROR_SPEC

   This object should be signed as original.

4. SENDER_TSPEC for SONET/SDH, G.709, or ETHERNET

   This object should be signed. It may be signed as immutable.

4.7.5 PathTear Message

In the PathTear message, the objects with end-to-end significance are:

1. CALL_ID

   This object needs to be signed to allow the Destination to determine who the Source is. It may be signed as immutable.

2. STYLE

   This object should be signed. It may be signed as immutable.

5. Data Structures, Error Handling, Logging, and Processing Rules

This section describes signatures that can be added to data items with end-to-end significance in UNI messages. It covers the structure of these signatures, rules for processing them, errors that can occur, and backward compatibility considerations.

This signature mechanism works at the RSVP application layer[1] of the OIF UNI. It is not defined for other UNI interfaces or protocols. It does not define any new RSVP message types to contain signatures, though it does extend the OIF UNI signaling model by it defining an end-to-end transport capability for user-defined information. It does use a private RSVP object OIF_VENDOR_PRIVATE_ EXTENSION_TYPE_3 defined in [PrivExt] and define an RSVP subobject OIF_E2E_SECURITY, which has its own Class Number and C-Type. This object and subobject are defined according to RSVP's rules for unknown and private-use Class Numbers and the OIF's conventions for using such private extensions [PrivExt]. Because of these rules:

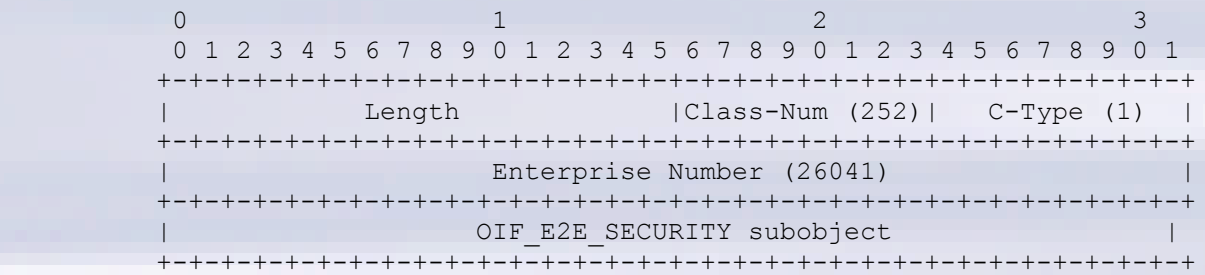
1. This service and mechanism are OPTIONAL to implement and optional to use. UNI-N and E-NNI implementations of this mechanism MUST be capable of configuring how to treat the E2E_UNI_SECURITY based on an enforceable network operator's policy. Such implementations MUST be configured, by default, so that they do not allow the E2E_UNI_SECURITY subobject to be used, so that, to use this mechanism with such implementations, it must be specifically enabled.
2. If this subobject arrives at UNI-N, E-NNI 1.0, or E-NNI 2.0 reference points as intermediate RSVP-TE-based protocol controllers on the path (i.e., UNI-C source to UNI-N destination), they should ignore it and forward it. As a matter of network operator's policy, however, UNI-N and E-NNI reference points may wish to

[1] Note that there is nothing essential about RSVP for the design of an end-to-end UNI signature mechanism. That is, all of the components could be defined as abstract objects and then instantiated in RSVP or other protocols. This was not done, because RSVP is the only protocol used in current OIF signaling. This type of generalization is left for future study if the need arises.

examine *all* message contents and apply whatever network operator's policy or misuse detection and responses are appropriate.

3. To work properly, this object and subobject need to be transported by both external and internal NNIs. (There is no requirement that all of the signaling data items remain intact or in the same format throughout internal processing.) Where this is not possible or not allowed, this mechanism will not work. As such, support for this mechanism is optional.
4. A UNI endpoint receiving this object and subobject and not recognizing them should, of course, ignore them. However, if it replies without a signature and policy requires one, it may then get an unexpected ResvErr message or Connection Release Request instead of a Connection Setup Confirm. The signaling operation should fail, and management plane intervention should be triggered.

The OIF_VENDOR_PRIVATE_ EXTENSION_TYPE_3 object contains not more than one subobject named OIF_E2E_SECURITY. It is shown for convenience below, although the authoritative definition for OIF_VENDOR_PRIVATE_ EXTENSION_TYPE_3 is in [PrivExt]:



The OIF_E2E_SECURITY subobject contains:

a. A subobject header (32 bits) with Length, Class-Num = 1, and C-Type = 1
b. A list of sub-subobjects containing the information listed in Section 4.1

Note: An alternative authentication mechanism, for future consideration, is to use a shared key and message authentication code instead of a signature. Using a signature has two distinct properties that would be lost in this case:

1. Only one party, the originating UNI-C, can create the signature
2. Any party, now or later, can verify the signature

End of Note.

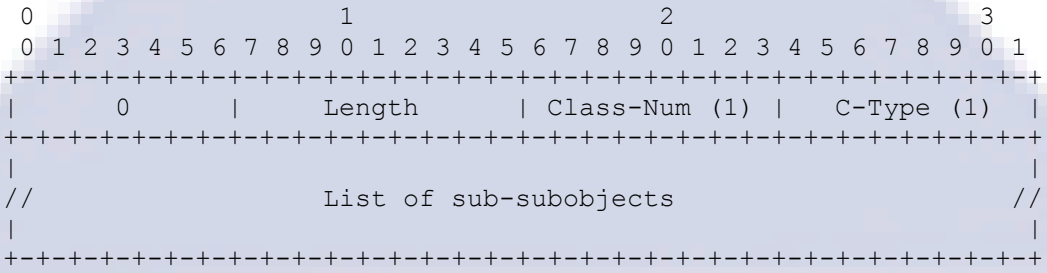
To facilitate implementation, the algorithms and data structures have been chosen to reuse constructions in:

- UNI 2.0 [UNI2.0-RSVP]
- The OIF's rules for RSVP Private Extensions [PrivExt]

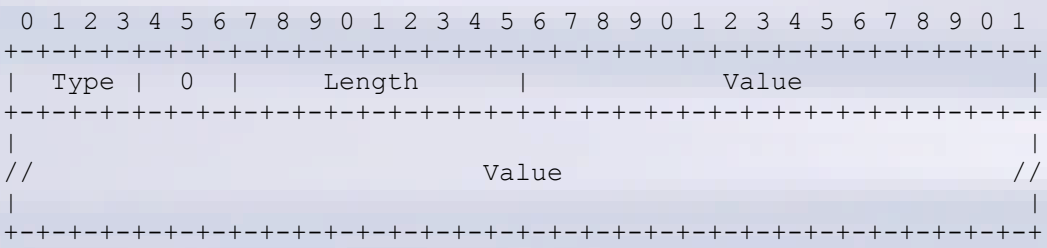
- IKEv2 [RFC5996]
- Syslog [RFC5424]

5.1 Structure of the OIF_E2E_SECURITY Subobject

This document defines one new RSVP subobject, OIF_E2E_SECURITY, of the OIF_VENDOR_PRIVATE_ EXTENSION_TYPE_3 object [PrivExt]:



The OIF_E2E_SECURITY subobject MUST NOT have Length (in the subobject header, above) greater than 256. The sub-subobjects all have the following format:

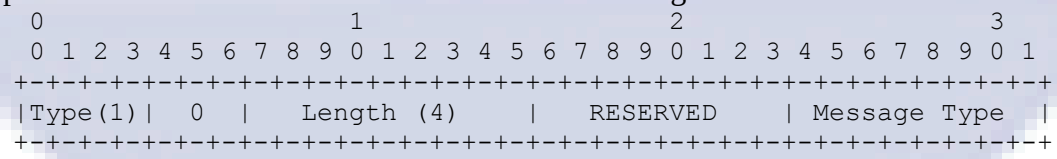


Length (in each sub-subobject header) contains the total length of the sub-subobject in bytes. It is an unsigned number and always a multiple of four.

The following sub-subobjects are defined. They SHOULD be included in the order listed here, but they MUST be accepted in any order. Fields labeled RESERVED SHOULD be set to all zeroes and MUST be ignored.

5.1.1 Message Type

This sub-subobject contains the Message Type in the original Common Header. It MUST be present and MUST NOT occur more than once. Length MUST be 4.



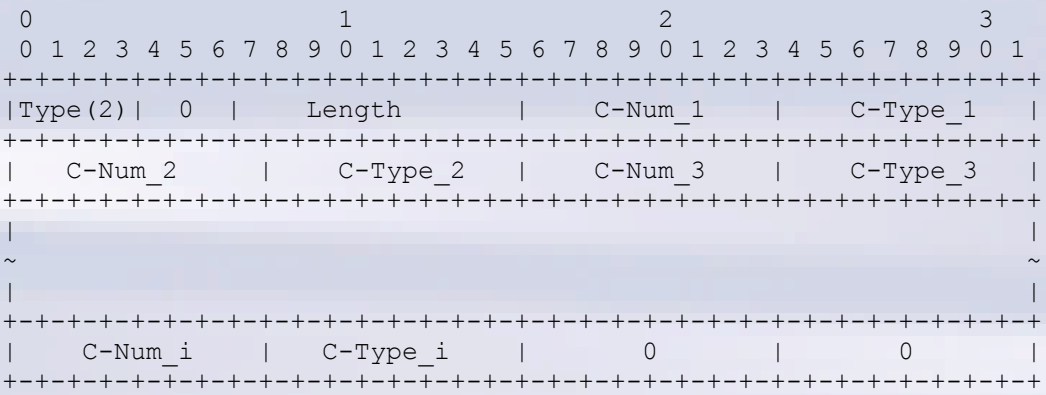
5.1.2 Immutable

This sub-subobject lists objects in the original message that must not change. It MUST occur zero or one time. It lists C-Num and C-Type pairs for which the outer RSVP message contains exactly one object. It indicates that this object is delivered unchanged. To compute and verify the signature, the corresponding objects are appended in the order listed to the signature block. If the C-Num and C-Type pair refers to the

OIF_VENDOR_PRIVATE_ EXTENSION_TYPE_3 object specified in [PrivExt], then the signature block is formed by removing the OIF_E2E_SECURITY subobject from this object and reducing its length accordingly.

Note 1: Any of the unique objects defined in [PrivExt] MAY be listed as immutable, subject to the adjustment of the OIF_VENDOR_PRIVATE_ EXTENSION_TYPE_3 object described here.

Note 2: No provision is included for listing any subobjects as immutable, including subobjects of the objects defined in [PrivExt]. If a need for such capability arises in future versions of the OIF UNI, extensions to this protocol to handle such cases can be considered.

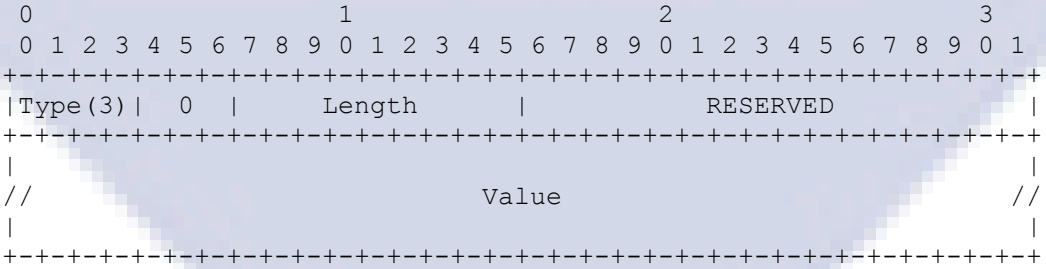


Length MUST be a multiple of 4. If one pair is listed, Length = 4; if two or three pairs are listed, Length = 8, and so on. If an even number of pairs is listed, the last two bytes are encoded as zeros.

5.1.3 Original

This sub-subobject replicates an object in the original message that may change. It MAY occur zero or more times.

Length is the length of the sub-subobject, a multiple of 4. This sub-subobject should be used only when an object may be changed, but its original value is important, for example, to verify that a null CALL_ID was filled in. Value is any complete object (i.e., a top-level object, neither a subobject nor a sub-TLV of a subobject) in the original message, zero padded to a multiple of four octets.



Note 1: If the original value of the OIF_VENDOR_PRIVATE_ EXTENSION_TYPE_3 is included, then implementations MUST remove the OIF_E2E_SECURITY subobject and adjust the length of the object accordingly.

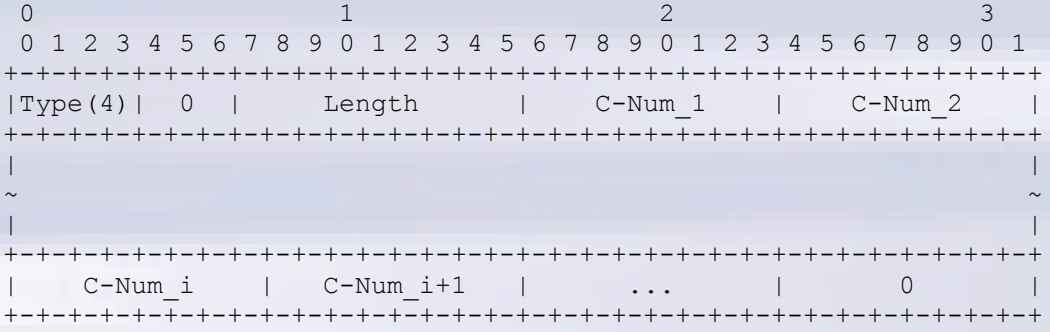
Note 2: If the signed original object contains one or more subobjects that will not be delivered to the destination UNI (e.g., because of processing rules or policy), then these subobjects MAY be removed from the signed copy of the original object. In this case, the size of the signed copy of the original object MUST be adjusted accordingly.

Note 3: No direct provision exists to include the original values of subobjects. If a need for such capability arises in future versions of the OIF UNI, extensions to this protocol to handle such cases can be considered.

5.1.4 C-Nums_Not_Present

This sub-subobject asserts that no object in the outer RSVP message had any of the C-Nums listed at the Source UNI. It MUST occur zero or one time.

Length MUST be a multiple of 4. For example, if one or two C-Nums are listed, Length = 4, if three through six are listed, Length = 8, and so forth. Unused entries in the last word are coded as 0.



Note 1: C-Num values should be listed only if they are important to identify as not present at the Source UNI as a matter of policy.

Note 2: No provision exists for identifying particular subobjects of an object as not present at the Source UNI.

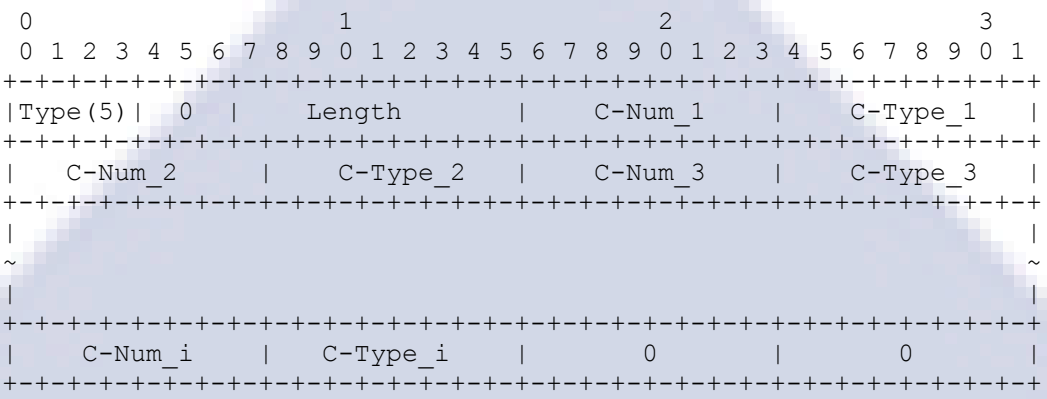
5.1.5 C-Nums_and_C-Types_Not_Present

This sub-subobject lists pairs of C-Num and C-Type values that it asserts did not occur for any object in the outer RSVP message at the Source UNI. It MUST occur zero or one time.

Length MUST be a multiple of 4. If one pair is listed, Length = 4; if two or three pairs are listed, Length = 8, and so on. If an even number of pairs is listed, the last two bytes are encoded as zeros.

Note 1: Pairs of C-Num and C-Type values should be listed only if they are important to identify as not present at the Source UNI as a matter of policy.

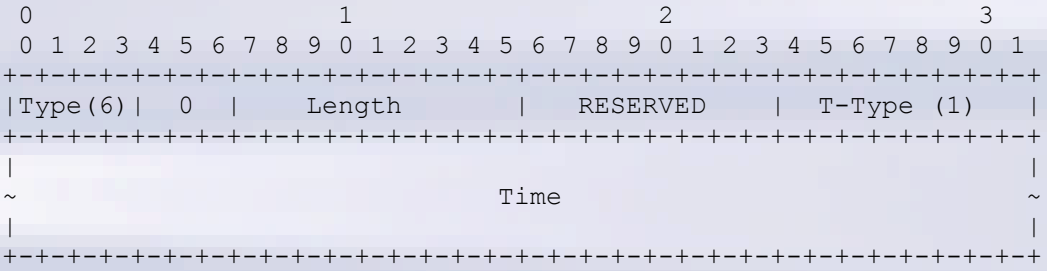
Note 2: No provision exists for identifying particular subobjects of an object as not present at the Source UNI.



5.1.6 Timestamp

This sub-subobject contains the time at which the message is sent. It is used to detect stale or replayed messages. It SHOULD be present and MUST NOT occur more than once. Implementations MUST verify the Timestamp sub-subobject when it is received.

One T-Type is defined for Timestamp, T-Type = 1. In this case, Time MUST be formatted as described in Section 6.2.3 of RFC 5424, The Syslog Protocol, [RFC5424], and padded with zeros as needed for 32-bit alignment. Length is 4 plus the number of bytes in the Time, excluding padding.



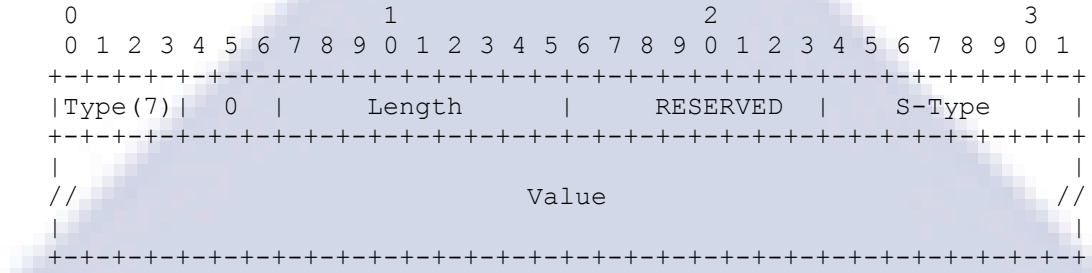
Implementations using the T-Type =1 TIMESTAMP sub-subobject MUST send a strictly increasing sequence of timestamps to each distinct receiver. The T-Type = 1 Timestamp sub-subobject has a maximum resolution of one microsecond, so implementations SHOULD have an accurate clock and SHOULD use the greatest precision available up to this limit. If another source of accurate time is not available, the Network Time Protocol (NTP) [RFC5905] is RECOMMENDED.

5.1.7 Signature

This sub-subobject contains the signature on the locally generated signature block. It MUST be present exactly once. S-Type corresponds to the IKEv2 Authentication Method (see [IANA-IKEv2]). S-Type = 3 (DSS with SHA-1) MUST be implemented.

A signature block is formed by starting with the OIF_E2E_SECURITY subobject, zeroing the Value field of this sub-subobject, and appending the immutable objects (starting with the Length, C-Num, and C-Type of each) to the OIF_E2E_SECURITY subobject in the

order listed in the Immutable sub-subobject. Then, the signature is computed over the signature block and inserted into the Value field.

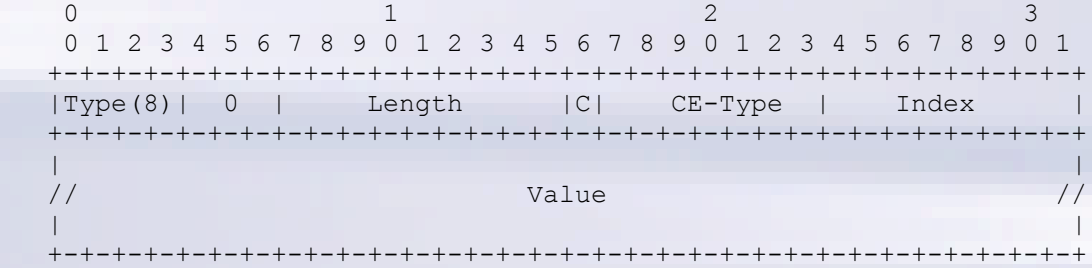


5.1.8 Cert_Encoding

This sub-subobject points to a certificate used for verifying the signature. It MUST occur at least once and MAY occur more than once. The first occurrence directs the verifier to the public key (i.e., the certificate encoding the public key) needed to verify the signature. Subsequent occurrences may be included to support this key (e.g., certificate chains).

If the C (cache) flag is 1, the receiver SHOULD store the sending UNI's TNA name (i.e., TNA name sub-type and value), the CE-Type, Index, and resulting certificate for future use. If the C flag is 0, the Length MUST be 4, the Value is omitted, and the triple <sending TNA name, CE-Type, Index> is used to locate the appropriate certificate. (The Index field provides ways for senders to avoid sending certificate pointers and to change certificates or for multiple senders to use the same TNA name with different certificates.)

If the sender sets the C flag to 1 and receives an error-free response, the sender SHOULD subsequently use the same CE-Type and Index and set the C flag to 0 when using this certificate with this receiver.



The following value for CE-Type is specified:

CE-Type = 1 The Value field contains an IKEv2 Certificate Payload as defined in Section 3.6 of [RFC5996] and beginning with a Cert Encoding byte. The hash-and-URL certificate (Cert Encoding = 12) MUST be implemented.

Notes to implementers:

- The TNA name may not actually be present in the message. Implementations need to keep a table of active CALL_IDs (i.e., call identifiers [RFC3474]) and their associated TNA names along with their sub-types to perform this operation. Implementations also need to include the local connection information (i.e., the UNI_IPv4_SESSION object) in this table.

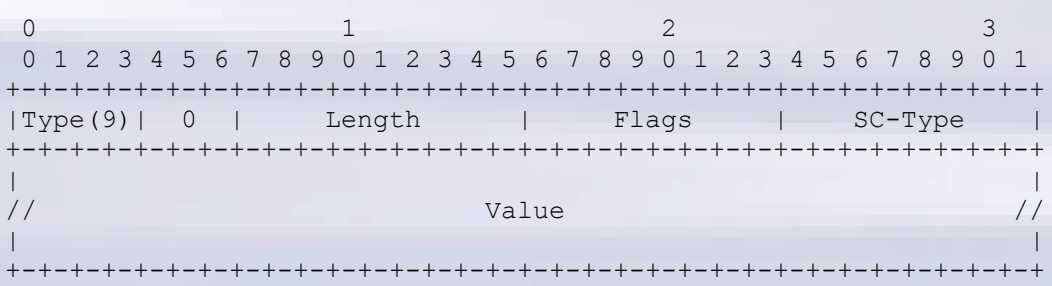
– Implementations may provide utilities apart from the signaling protocol to help users populate and maintain certificate caches.

– If the C flag is 0 and the certificate is not available, an error has occurred. If certificates are not being sent but are managed out of band, implementations MAY try to obtain the certificate by other means before returning an error. In any case, implementations MAY return the error indication OIF_E2E_SECURITY_NO_KEY. Local policy determines whether the Path_State_Removed flag is set.

  If the Value field points the receiver to a certificate that must be retrieved, then the certificate itself SHOULD be stored to avoid repeated lookups.

5.1.9 Security_Credentials

This sub-subobject contains additional security information. It MAY occur zero or more times.



The following values are defined for SC-Type:

- 1 OCSP

  A Flags value of 1 indicates a request for OCSPResponse data in any signed replies. Otherwise, Flags MUST be 0. The Value, if present, contains a DER-encoded OCSPResponse as defined in RFC 2560 [RFC2560]. This sub-subobject MUST NOT occur more than once with a SC-Type 1 and Flags value of 1.

- 2 IV

  The Value specifies one or more initialization vectors needed for securing end-to-end data (e.g., Ethernet or SONET/SDH payload data) in the data plane.

- 3 Synch

  The Value specifies synchronization information needed for securing end-to-end data (e.g., Ethernet or SONET/SDH payload data) in the data plane.

- 4 KA

  The Value specifies key agreement material needed for securing end-to-end data (e.g., Ethernet or SONET/SDH payload data) in the data plane.

5.2 Error Codes and Error Logging

To configure this signature mechanism for the OIF UNI, users need to carry out a sequence of processes:

1. Ensure that their service provider allows the use of this mechanism
2. Generate signing keys and obtain a certificates for the corresponding verification keys
3. Distribute their certificates to other UNI-C's that will need them
4. Set up security policy, which includes deciding which incoming and outgoing messages will be signed and how
5. Verify which objects will be protected as original or immutable, and make sure the resulting message lengths will be acceptable
6. Verify that suitable signaling channel throughput and performance will be available
7. Set up logging for security messages

Implementers should provide tools to carry out these operations and check that they are done completely and consistently. If these processes are carried out interactively, a user interface is needed, and various warnings or errors need to be presented, explained, and addressed. If these processes are carried out automatically, then warning or error conditions should be logged with the appropriate severity needed to generate the necessary alarms. In either case, how these processes work and how these errors are presented and handled are left to implementers. They do not involve any protocol operations or interoperability considerations. Considering common management interfaces to such functions is a matter for future study.

This IA focuses on the error conditions that occur when the OIF_E2E_SECURITY subobject is actually sent, received, and processed, or when it is expected but not received. Each step in the receiver's processing rules is associated with an identifiable error condition, and this IA explains how to record and respond to each of these conditions. Especially because this IA uses protocols to define a *security* mechanism, the mechanism defined here needs to balance, on the one hand, providing enough tools to debug and diagnose the mechanism against, on the other hand, revealing too much information in response to hostile probes.

These errors SHOULD be recorded in a log when they are generated or received. They provide specific information as to what has gone wrong. Because these errors may result from attacks on the protocol and sending specific error messages to an attacker may be undesirable, implementers SHOULD provide users with a way to specify in their security policies how errors are handled.

The strictest approach is to provide the attacker with no information at all, that is, do not even reply. Sending OIF_E2E_SECURITY_UNSPECIFIED_ERROR message in all cases is a more forgiving approach. At the other extreme, one could simply record errors and continue processing, which might be suitable for experimenting with the security mechanism. One approach is to define four levels for error handling, from strictest to most lenient:

1. Silently discard the message. This provides an attacker with no indication as to what has happened.
2. Send a generic error reply (OIF_E2E_SECURITY_UNSPECIFIED_ERROR). This provides an attacker with minimal information.
3. Send a specific error reply. This may provide an attacker with specific information.
4. Note the error and ignore it; accept the message and continue processing as though the message were valid. This may allow an attacker to forge or modify messages.

The user-defined error message as defined in [RFC5284] is used according to the rules in [PrivExt] when sending error replies resulting from using the OIF_E2E_SECURITY object. To specify a user-defined error message, the standard ERROR_SPEC object (Class = 6) is sent with the error code 33 (User Error Spec) and error value 0. The USER_ERROR_SPEC object (Class=194, C-Type=1) MUST be sent when the ERROR_SPEC error code is set to 33. The fields of the USER_ERROR_SPEC object are set as follows:

- Enterprise Number = 26041 (i.e., OIF)
- Sub Org = 1
- Err Desc Len = 0 (no error description)
- User Error Value as defined below
- Err Desc = Null (not present)
- User Defined Subobjects (not present)

As defined in Section 9.1 of [UNI2.0-RSVP], this object MAY be included in a PathErr (Section 9.1.4) or ResvErr (Section 9.1.8) message, as appropriate.

Notes to implementers:

- Implementations MAY use the OIF_E2E_SECURITY subobject to sign these ERROR_SPEC and USER_ERROR_SPEC objects (as well as other items as appropriate) in these error replies. The ERROR_SPEC object and the USER_ERROR_SPEC object SHOULD be signed as Immutable.
- Implementations MUST NOT reply to these error messages with another error message.

- Implementations using this end-to-end authentication mechanism and receiving an unauthenticated error message SHOULD anticipate that the error message may be a denial of service attack and allow time for a legitimate response before acting on the reported error.

Seven User Error Values are defined and used as follows:

- OIF_E2E_SECURITY_REQUIRED, User Error Value = 1

  Policy requires an OIF_E2E_SECURITY subobject for this message, but none was received, or policy requires that certain objects in the message be signed but they were not.

  If generators of this error message use logging [LogAud], they SHOULD log this error message with:

  – SD-ID E2E_SEC@26041
  – SEVERITY 3 (Error)
  – SD parameters TYPE="<rsvp message type>"
  – ERROR="SECURITY_REQUIRED", DIR="T", and also the CALL_ID and TNA names along with their sub-types for both parties as appropriate

  If receivers of this error message use logging [LogAud], they SHOULD log this error message with:

  – SD-ID E2E_SEC@26041
  – SEVERITY 3 (Error)
  – SD parameters TYPE="<rsvp message type>", ERROR="SECURITY_REQUIRED", DIR="R", and also the CALL_ID and TNA names along with their sub-types for both parties as appropriate

- OIF_E2E_SECURITY_NO_KEY, User Error Value = 2

  This error message MAY be sent if a key to verify the signature cannot be obtained.

  If generators of this error message use logging [LogAud], they SHOULD log this error message with:

  – SD-ID E2E_SEC@26041
  – SEVERITY 3 (Error)
  – SD parameters TYPE="<rsvp message type>", ERROR="NO_KEY", DIR="T", and also the CALL_ID and TNA names along with their sub-types for both parties as appropriate

  If receivers of this error message use logging [LogAud], they SHOULD log this error message with:

  - SD-ID E2E_SEC@26041
  - SEVERITY 3 (Error)
  - SD parameters TYPE="<rsvp message type>", ERROR="NO_KEY", DIR="R", and also the CALL_ID and TNA names along with their sub-types for both parties as appropriate

- OIF_E2E_SECURITY_INVALID_CERT, User Error Value = 3

  A certificate containing the signature verification key or needed to obtain the signature verification key has some problem, e.g., it may have expired or been revoked.

  If generators of this error message use logging [LogAud], they SHOULD log this error message with:

  - SD-ID E2E_SEC@26041
  - SEVERITY 3 (Error)
  - SD parameters TYPE="<rsvp message type>", ERROR="INVALID_CERT", DIR="T", NAME="<name in the certificate>", and also the CALL_ID and TNA names along with their sub-types for both parties as appropriate

  If receivers of this error message use logging [LogAud], they SHOULD log this error message with:

  - SD-ID E2E_SEC@26041
  - SEVERITY 3 (Error)
  - SD parameters TYPE="<rsvp message type>", ERROR="INVALID_CERT", DIR="R", NAME="<name in the certificate>", and also the CALL_ID and TNA names along with their sub-types for both parties as appropriate

- OIF_E2E_SECURITY_UNAUTHORIZED_SIGNER, User Error Value = 4

  Policy specifies that the name in the certificate is not permitted to send this message.

  If generators of this error message use logging [LogAud], they SHOULD log this error message with:

  - SD-ID E2E_SEC@26041
  - SEVERITY 3 (Error)
  - SD parameters TYPE="<rsvp message type>", ERROR="UNAUTHORIZED_SIGNER", DIR="T", NAME="<name in the certificate>", and also the CALL_ID and TNA names along with their sub-types for both parties as appropriate

If receivers of this error message use logging [LogAud], they SHOULD log this error message with:

- SD-ID E2E_SEC@26041
- SEVERITY 3 (Error)
- SD parameters TYPE="<rsvp message type>", ERROR="UNAUTHORIZED_SIGNER", DIR="R", NAME="<name in the certificate>", and also the CALL_ID and TNA names along with their sub-types for both parties as appropriate

- OIF_E2E_SECURITY_INVALID_SIGNATURE, User Error Value = 5

  The certificate is valid, but the signature verification failed.

  If generators of this error message use logging [LogAud], they SHOULD log this error message with:

  - SD-ID E2E_SEC@26041
  - SEVERITY 3 (Error)
  - SD parameters TYPE="<rsvp message type>", ERROR="INVALID_SIGNATURE", DIR="T", NAME="<name in the certificate>", and also the CALL_ID and TNA names along with their sub-types for both parties as appropriate

  Also, generators of this error message that use logging SHOULD log the entire message with the PROT@26041 message as described in [LogAud].

  If receivers of this error message use logging [LogAud], they SHOULD log this error message with:

  - SD-ID E2E_SEC@26041
  - SEVERITY 3 (Error)
  - SD parameters TYPE="<rsvp message type>", ERROR="INVALID_SIGNATURE", DIR="R", NAME="<name in the certificate>", and also the CALL_ID and TNA names along with their sub-types for both parties as appropriate

- OIF_E2E_SECURITY_CONTENT_ERROR, User Error Value = 6

  The certificate is valid, the signature verified, but either the contents of the outer RSVP message do not correspond to what the OIF_E2E_SECURITY object asserts or the Timestamp is stale.

  If generators of this error message use logging [LogAud], they SHOULD log this error message with:

  - SD-ID E2E_SEC@26041

- SEVERITY 3 (Error)
- SD parameters TYPE="<rsvp message type>", ERROR="CONTENT_ERROR", DIR="T", and OPTIONALLY the CALL_ID and TNA names along with their sub-types for both parties as appropriate

Also, generators of this error message that use logging SHOULD log the entire message with the PROT@26041 message as described in [LogAud].

If receivers of this error message use logging [LogAud], they SHOULD log this error message with:

- SD-ID E2E_SEC@26041
- SEVERITY 3 (Error)
- SD parameters TYPE="<rsvp message type>", ERROR="CONTENT_ERROR", DIR="R", and OPTIONALLY the CALL_ID and TNA names along with their sub-types for both parties party as appropriate

- OIF_E2E_SECURITY_UNSPECIFIED_ERROR, User Error Value = 7

  An OIF_E2E_SECURITY error occurred that (1) cannot be categorized by other error codes or (2) is left unspecified for policy reasons.

  This error can occur if more than one OIF_E2E_SECURITY subobject is received, or if the OIF_E2E_SECURITY subobject is improperly formatted.

  If generators of this error message use logging [LogAud], they SHOULD log this error message with:

  - SD-ID E2E_SEC@26041
  - SEVERITY 3 (Error)
  - SD parameters TYPE="<rsvp message type>", ERROR="UNSPECIFIED_ERROR", DIR="T", and the CALL_ID and TNA names along with their sub-types for both parties as appropriate

  Also, generators of this error message that use logging SHOULD log the entire message with the PROT@26041 message as described in [LogAud].

  If receivers of this error message use logging [LogAud], they SHOULD log this error message with:

  - SD-ID E2E_SEC@26041
  - SEVERITY 3 (Error)
  - SD parameters TYPE="<rsvp message type>", ERROR="UNSPECIFIED_ERROR", DIR="R", and the CALL_ID and TNA names along with their sub-types for both parties as appropriate

5.3 Processing Rules

The OIF_E2E_SECURITY subobject MAY be included in messages with end-to-end significance. These include Path, Resv, ResvConf, PathErr, ResvErr, and PathTear. It MUST NOT be used in messages without end-to-end significance (e.g., Hello, Ack, Bundle, Notify, and Srefresh).

Implementations SHOULD be careful to use the features described herein in a way that minimizes the size of the OIF_E2E_SECURITY subobject.

Implementations MUST include a method for users to specify an end-to-end security policy that includes specifying:

- For each other UNI-C (identified by a TNA name along with its sub-type) using the OIF_E2E_SECURITY subobject, the RSVP message types and objects requiring the OIF_E2E_SECURITY subobject (e.g., GENERALIZED_UNI in a Path message)
- The acceptable criteria for signing certificates
- How OIF_E2E_SECURITY errors are handled

For active calls, implementations MUST also know how to map messages to TNA names.

The following steps illustrate how to generate and check messages with the OIF_E2E_SECURITY subobject in them.

For senders:

1. Determine whether the OIF_E2E_SECURITY subobject is required for this message and, if so, continue.
2. Obtain the appropriate signing key.
3. Generate the RSVP message with zeros in the Signature Value.
4. Copy the Message Type into the OIF_E2E_SECURITY subobject.
5. Generate a TIMESTAMP subobject with a later time than any previously sent to this receiver.
6. Determine which objects need to be signed.
7. Form a signature block that has immutable objects appended.
8. Calculate the signature over the signature block and insert it into the OIF_E2E_SECURITY subobject.
9. Calculate and insert the checksum in the outer RSVP Common Header.

For receivers:

1. Verify the checksum in the outer RSVP Common Header.

2. Determine according to policy whether the OIF_E2E_SECURITY subobject is required for this message. If so, and the OIF_E2E_SECURITY subobject is not present, record the error OIF_E2E_SECURITY_REQUIRED. If not, discard any OIF_E2E_SECURITY subobject and skip the remaining steps.
3. Check that the signing key is appropriate and valid. If not, record the error OIF_E2E_SECURITY_UNAUTHORIZED_SIGNER and exit.
4. Retrieve the verification key. If not possible, record the error OIF_E2E_SECURITY_NO_KEY and exit.
5. Form the signature block as above and verify the signature. If this fails, record the error OIF_E2E_SECURITY_INVALID_SIGNATURE and exit.
6. Check according to policy that all objects that are required to be signed actually are. If not, record the error OIF_E2E_SECURITY_REQUIRED and exit.
7. Check that the outer RSVP message corresponds appropriately to what was signed. Check that the Timestamp is later than any previously received from this sender. (Implementations MAY track clock skew and round-trip times for future reference.) If any of this fails, record the error OIF_E2E_SECURITY_CONTENT_ERROR and exit.
8. Record any auxiliary Security_Credentials information for later use.
9. Update dynamic policy tables as needed.

5.4 Backward Compatibility

This section describes what happens when a first UNI-C attempts to use the OIF_E2E_SECURITY subobject but other OIF reference points on the UNI-C to UNI-C end-to-end path are unaware of it.

If the other UNI endpoint does not recognize the OIF_E2E_SECURITY subobject, it should, according to [PrivExt], ignore it. If the first UNI-C does not require this subobject in return, signaling may work normally, as if no signatures are being used. If the first UNI-C does, however, expect a valid signature in return, this is a configuration error, and the response without a signature should be rejected.

At intermediate protocol controllers between a UNI-C and UNI-N running an RSVP-TE session, the processing rules in [PrivExt] and for RSVP-TE in general state that this sub-subobject should be passed on unchanged. If I-NNIs or E-NNIs upstream of the UNI-N are based on RSVP-TE and the operator's policy allows use of the OIF_E2E_SECURITY subobject, it needs to be transported (i.e., interworked or mapped) over all such I-NNI and E-NNI sessions. However, at a UNI-N or E-NNI reference point a service operator may enforce a network operator's policy that prohibits unrecognized objects, unrecognized subobjects, or malformed RSVP-TE messages in general. In this case, messages containing the OIF_E2E_SECURITY subobject may be rejected. Note, however, that implementations unaware of this subobject will also be unaware of the requirement that it must be delivered and installed in a disabled state.

At an intermediate I-NNI running a signaling protocol other than RSVP-TE, the handling of the OIF_E2E_SECURITY sub-subobject is unspecified. If this reference point is configured to follow the processing rules in [PrivExt] and does not apply any filtering rules, it may arrange to communicate the OIF_E2E_SECURITY sub-subobject in a way that it can be reconstructed at the other RSVP-TE UNI endpoint. In other cases, it may not do this, and this mechanism will not work as presently defined.

6. Policy Considerations at UNI Endpoints

6.1 Specifying, Enforcing, and Changing Policies without Disruption

Security is not much use without an enforceable security policy. An attacker could simply remove the signature from a message and modify it: the receiver would not know that the original message was ever signed.

Policy includes who the other party is (TNA name), what (message types and objects) needs to be protected, how the protection is applied (e.g., what cryptographic methods, keys, and replay counters to use), and how errors are handled. Security policies are needed both for sending and receiving signed messages. Senders consult policy to decide what protection to apply to outgoing messages, and receivers do the same to determine what protection is required for incoming messages.

Therefore, implementations MUST support a security policy that lets users specify which messages and objects must be signed with which keys.

One approach is to start with a global static policy table indexed by TNA name that describes incoming and outgoing security policy for each other TNA name with which they use the OIF_E2E_SECURITY subobject and the corresponding certificates and keys. At the top level, the choices for each TNA name are (1) allow unsecured calls; (2) require security on all calls; or (3) block all calls. Conventions for specifying ranges of TNA name or default policy may be included.

Next, implementations may maintain a table of active CALL_IDs using the OIF_E2E_SECURITY subobject with local connection information as needed and pointers to the table of parties. Separate tables of incoming and outgoing security policy for each active call might be used. These may be indexed by CALL_ID.

Finally, a sorted list of local connection information with pointers to CALL_IDs may be maintained.

Policy MUST specify how errors are handled, as described in Section 5.2.

It may be important for auditing or such purposes to keep track of past policies and policy changes. Logging of policy changes and who made them is RECOMMENDED.

Vendors should provide tools for setting up and maintaining security policies. In fact, such tools should let users align policies at both ends of a potential call and check that this has been done correctly. The tools may also allow users to schedule changes in policies in a coordinated way. Vendors may decide how the user interface to these tools works and whether the same set of tools should perhaps deal with certificate distribution.

During normal processing, upon restart, or after signaling channel recovery, users may wish to change security policies. It is important that security policy changes do not cause unexpected control plane or data plane behavior, particularly for existing calls. Users should test security policies in a safe environment before applying them to live traffic, coordinate policy changes at both ends of an existing or planned call, follow a safe order of applying policy changes, and log the results for later examination. Implementations SHOULD provide tools and warnings to help users update policies safely and to avoid errors. It may be useful to allow scheduling of policy changes or even to allow policies to vary regularly according to calendar or time of day. Two RECOMMENDED capabilities are (1) to allow users to specify whether policy changes apply to all new and existing calls, new calls and certain existing calls, or just new calls, and (2) to allow users to schedule policy changes to go into effect at a certain scheduled time.

Policy enforcement has to be turned on or off in a safe sequence to avoid errors. For example, to raise the required security level:

1. Set the receiver to accept the higher security level.
2. Set the sender to apply this higher security level.
3. Set the receiver to require the higher security level.

To lower the required security level, these steps can be reversed.

6.2 Signaling Channel Failure, Restart, and Policy Updates

Recovery and restart need to take into account both changes in security policy and changes in state.

When the signaling channel fails and then is restarted, a UNI-C may have missed messages, some of which contained signatures.

Section 8.14 of [UNI2.0-RSVP] describes how a UNI-C and UNI-N resume operations when recovering from a failure of one party or loss of signaling connectivity. Implementations MUST follow the restart procedures in [UNI2.0-RSVP].

UNI 2.0 restart procedures may result in a change of state (i.e., removed connections) without delivering the signaling messages that caused the change of state. Users need to understand that, policy for securing such lost messages notwithstanding, the UNI-C may need to accept such state changes. If logging is used, details of such removed connections MUST be logged with:

- SD-ID E2E_SEC@26041
- SEVERITY 4 (Warning)
- SD parameters TYPE= “Srefresh: connection removed”, and the CALL_ID and TNA names along with their sub-types for both parties as appropriate

A UNI-N SHOULD attempt to ensure that a signature compliant with the network operator’s policy is delivered at least once. If the signaling protocol is unreliable, and there is reason to believe that the signature may not have been received, it may be sent

repeatedly. As with all unreliable protocols, it may be a configurable option to send a message including a signature a certain number of times or every time a corresponding refresh message is sent. The graceful restart and recovery procedure should resend any signatures that may not have been received. A UNI-C, upon receiving a duplicate copy of a message with a signature, MUST ignore the retransmitted signature.

In the unlikely event that a signature is required and not received, the signaling operation may result in an error and need to be repeated. This may, of course, occur with any signaling message transmitted with an unreliable protocol.

The following items have to be maintained across restarts:

- Keying material
- Certificate caches
- Policy tables
- TNA name to CALL_ID mappings for existing connections
- Replay counters

The remaining aspects of this mechanism should be stateless.

7. OIF Assigned Numbers

Within the OIF_VENDOR_PRIVATE_ EXTENSION_TYPE_3 object [PrivExt], Class Number 1 with C-Type 1 is reserved for the OIF_E2E_SECURITY subobject.

When using RSVP Class Number 194 and C-Type = 1 with the OIF's Enterprise number 26041, Sub Org = 1 is reserved for errors resulting from using the OIF_E2E_SECURITY object (see [PrivExt] and [RFC5284]).

8. Performance

This mechanism entails some overhead in processing cycles and message size.

Open source software for the cryptographic functions is available. For a light load of say a few messages a second, the overhead is quite manageable on today's microprocessors. For heavier loads, there are well-developed technologies for high volume cryptography (e.g., those developed for secure web servers), which can handle the constructions defined for this signature mechanism.

Commonly expected use of this signature mechanism should be achievable with less than 150 bytes of message expansion or perhaps a little more in some cases. Normal use should not cause fragmentation or other undesirable effects. A strict limit of 256 bytes has been placed on the length of the E2E_UNI_SECURITY subobject. This limit should exceed what ordinarily occurs. A signaling communications network supporting end-to-end UNI authentication needs to be designed to account for this.

The first Path message in a call may be a worst-case example over all message types. With certificate caching and 32-bit TNA names, the overhead may be 144 bytes, as shown in

Table 1. The size limit of 256, in this case, still allows sufficient room for a longer Cert_Encoding sub-subobject (Section 5.1.8) and an additional Security_Credentials sub-subobject (Section 5.1.9).

Item	Length
OIF_VENDOR_PRIVATE_ EXTENSION_TYPE_3 header	8 bytes
OIF_E2E_SECURITY header	4 bytes
Message Type sub-subobject	4 bytes
List of four immutable objects (ADMIN_STATUS, GENERALIZED _LABEL_REQUEST, SENDER_TSPEC, SESSION_ATTRIBUTE)	12 bytes
Original CALL_ID object	12 bytes
Original GENERALIZED_ UNI_ATTRIBUTES object	20 bytes
Timestamp sub-subobject	36 bytes
Signature sub-subobject	44 bytes
Cert_Encoding sub-subobject	4 bytes

Table 1: Example of Message Size for Signing a Path Message.

9. Security Considerations

Both the DSS signatures and the hash-and-URL certificate lookup rely on the collision resistance of the SHA-1 hash function. Stronger hash functions for these methods should be considered if and when they are standardized by the IETF or SHA-1 is officially declared broken. A potentially stronger combination would be to use SHA-256 and the elliptic curve digital signature algorithm (ECDSA) with a signature of corresponding length. Algorithm agility is provided in the data structures, and implementations should allow for this type of enhancement in the future.

Signers using DSS must protect not only their private keys but also *all* of the random numbers used to generate signatures. Thus, signers must have a strong method for generating pseudo-random numbers. See Section 4.1 for references on this topic. Some implementations of DSS have been shown to be insecure because implementers did not enforce range checks in the specification.

The mechanism described in this IA imposes overhead on a service provider's signaling communications network and allows end-to-end communications between users. These communications are designed to support the integrity of the signaling and security for the transport resources set up by signaling. Service providers should enforce appropriate use of this mechanism based on their policies. In particular, they may allow this mechanism, prohibit it, or place limits on its use. They may, for example, limit it by any combination of number of occurrences, aggregate overhead, or per-message overhead. Service providers

may choose to drop non-conforming traffic, respond with an error condition, or remove the non-conforming parts of messages.

Future signaling extensions may, of course, impact the way this mechanism is defined and used. Therefore, this IA needs to be maintained along with new OIF developments in signaling. For example, a future signaling enhancement may be to allow the network to translate the DESTINATION_TNA. If this data item is signed as Original, the destination UNI may verify the signature but find two different values. In this hypothetical example, the destination UNI may choose to check whether the translation done by the network was appropriate or not.

A denial of service attack carried out by flooding a UNI receiver with a large number of invalid signatures is possible. Upstream ingress and egress filtering can be used to block off-path attacks of this sort. If this is an insufficient or impractical remedy, then, the methods in [SecExt] may be used to filter such attacks more efficiently.

When possible, error responses defined in this IA should include a signature to prevent denial of service attacks based on forging these error messages.

10. Short Signatures (Informative)

This section is included only for future consideration. It does not provide any guidance for this Implementation Agreement. Also, this section offers no opinions about intellectual property considerations.

One of the main design goals in this IA is to keep the size of the OIF_E2E_SEC subobject as small as possible. The DSS signature, which is 320 bits or 40 bytes, is a necessary part of this subobject, so it is worthwhile to consider shorter alternatives. Three approaches have been proposed:

- Use signatures with message recovery. If the data that are signed contain part of the message itself, then the inverse verification operation can recover these bits, and the effective overhead of the signature is reduced accordingly.
- Use signatures based not on computational number theory but rather on multivariate cryptography, coding theory, or lattices.
- Use bilinear pairings, which allow digital signatures based on discrete logarithms to be expressed with one, say, 160-bit parameter instead of two.

Naïve approaches to message recovery have pitfalls. For instance, signing the actual message with textbook RSA allows an attacker to obtain existential forgeries of signatures, so, as always, constructions need to be analyzed carefully and not invented in an ad hoc fashion.

In 2000, Naccache and Stern [NS00] described a partial message recovery scheme that shortens 40-byte DSS signatures (or the analogous construction based on elliptic curves) to 26 bytes with no loss in security. They provide a security proof in the random oracle model.

In 2001, Patarin, Courtois, and Goubin first presented QUARTZ [PCG01], a 128-bit signature scheme based on multivariate polynomials. Signing a message, however, is slow. Potential users should also refer to the most up-to-date information about cryptanalysis of such schemes.

Also in 2001, Boneh, Lynn, and Shacham [BLS01] used a totally different approach, bilinear pairings on an elliptic (or hyper-elliptic) curve, to derive a short signature scheme based on the computational Diffie-Hellman assumption and proved its security in the random oracle model. It uses the private key extraction technique in the Boneh-Franklin identity-based encryption scheme to eliminate the need for two 160-bit parameters and reduce the size of the corresponding signature to 20 bytes. However, it requires a special kind of hash function called PointToMap.

In 2004, Zhang, Safavi-Naini, and Susilo [ZSS04] showed how to construct a more efficient short signature scheme based on bilinear pairings that can use any hash function (such as SHA-1). Their security proof is based on the inverse computational Diffie-Hellman assumption and also works in the random oracle model.

Also in 2004, Boneh and Boyen [BB04] described efficiency and security improvements to their short signature scheme based on bilinear pairings. Their newer security proof avoids using a random oracle but relies on the strong Diffie-Hellman assumption.

11. Summary

This Implementation Agreement defines an optional OIF UNI extension to provide end-to-end authentication of signaling messages between two OIF UNI 2.0 clients by transporting a signature on a subset of the objects in signaling the message end to end. Support for this end-to-end transport is optional.

12. References

12.1 Normative References

The following references contain provisions that, through reference in this text, constitute provisions of this IA. At the time of publication, the versions indicated were current and valid. Many references are subject to revision, and parties to agreements based on this IA are encouraged to investigate the possibility of applying the most recent versions of the references indicated below.

[FIPS186-3] National Institute of Standards and Technology Federal Information Processing Standard 186-3, *Digital Signature Standard (DSS)*, June 2009.

[IANA-IKEv2] http://www.iana.org/assignments/ikev2-parameters

[LogAud] Optical Internetworking Forum Implementation Agreement OIF-SLG-01.2, “OIF Control Plane Logging and Auditing with Syslog version 1.1,” November 2011.

[PrivExt] Optical Internetworking Forum Implementation Agreement RSVP-PVT-EXT-01.0, “OIF Application of Vendor Private Extensions in RSVP,” October 2011.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," IETF RFC 2119, March 1997.

[RFC2205] Braden, R., et al., "Resource ReSerVation Protocol (RSVP) --Version 1 Functional Specification," IETF RFC 2205, September 1997.

[RFC2560] Myers, M., et al., "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol – OCSP," IETF RFC 2560, June 1999. See also RFC 5019, draft-ietf-pkix-rfc2560bis, and RFC 6277.

[RFC3474] Lin, Z., and D. Pendarakis, "Documentation of IANA assignments for Generalized MultiProtocol Label Switching (GMPLS) Resource Reservation Protocol - Traffic Engineering (RSVP-TE) Usage and Extensions for Automatically Switched Optical Network (ASON)," IETF RFC 3474, March 2003.

[RFC5280] Cooper, D., et al., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," IETF RFC 5280, May 2008.

[RFC5284] Swallow, G., and A. Farrel, "User-Defined Errors for RSVP," IETF RFC 5284, August 2008.

[RFC5424] Gerhards, R., "The syslog Protocol," IETF RFC 5424, March 2009.

[RFC5905] Mills, D., et al., "Network Time Protocol Version 4: Protocol and Algorithms Specification," IETF RFC 5905, June 2010.

[RFC5996] Kaufman, C., P. Hoffman, Y. Nir, and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)," IETF RFC 5996, September 2010.

[SecExt] Optical Internetworking Forum Implementation Agreement OIF-SEP-03.1, "Security Extension for UNI and NNI version 2.0," November 2011.

[UNI2.0] OIF Implementation Agreement, "User Network Interface (UNI) 2.0 Signaling Specification Common Part," OIF-UNI-02.0-Common, February 2008.

[UNI2.0-RSVP] OIF Implementation Agreement, "User Network Interface (UNI) 2.0 Signaling Specification OIF-UNI-02.0-RSVP - RSVP Extensions for User Network Interface (UNI) 2.0 Signaling," OIF-UNI-02.0-RSVP, February 2008.

[X.509] ITU-T Recommendation X.509: Information technology - Open Systems Interconnection - The Directory: Public-key and attribute certificate frameworks, August 2005.

12.2 Informative References

[BFW11] Behringer, M., F. Le Faucheur, and B. Weis, "Applicability of Keying Methods for RSVP Security," IETF RFC 6411, October 2011.

[BB04] Boneh, D., and X. Boyen, "Short signatures without random oracles and the SDH assumption in bilinear groups," in *J.Cryptology* 21(2), 2008, pp. 149-177. Preliminary version in C. Cachin and J. Camenisch, eds., *Eurocrypt 2004*, Springer LNCS vol. 3027, pp. 56–73.

[BLS01] Boneh, D., B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," In *J. Cryptology*, 17(4):297–319, 2004. Extended abstract in C. Boyd, ed., *Asiacrypt 2001*, Springer LNCS vol. 2248, 2001, pp. 524–532.

[CarrierReq] OIF Carrier WG Guideline Document: Control Plane Requirements for Multi-Domain Optical Transport Networks, CWG # OIF-CWG-CPR-01.0, July 2010.

[E-NNI] Optical Internetworking Forum Implementation Agreement, "OIF E-NNI Signaling Specification," OIF-E-NNI-Sig-02.0, 2009.

[Gut98] Gutmann, P., "Software Generation of Practically Strong Random Numbers," *Seventh USENIX Security Symposium Proceedings*, The USENIX Association, 1998, pp. 243–257.

[Katz] Katz, J., *Digital Signatures*, Springer-Verlag, 2010.

[KSF99] Kelsey, J., B. Schneier, and N. Ferguson, "Notes on the Design and Analysis of the Yarrow Cryptographic Pseudorandom Number Generator," *Sixth Annual Workshop on Selected Areas in Cryptography*, Springer-Verlag, 1999.

[Koç09] Koç, Ç., ed., *Cryptographic Engineering*, Springer-Verlag, 2009.

[NS00] Naccache, D., and J. Stern, "Signing on a Postcard," In Y. Frankel, ed., *Financial Cryptography*, Springer LNCS vol. 1962, 2000, pp. 121–135.

[NIST800-90] Barker, E., and J. Kelsey, *Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised)*, NIST Special Publication 800-90, March 2007.

[PCG01] Patarin, J., N. Courtois, and L. Goubin, "QUARTZ, 128-bit long digital signatures," in *CT-RSA 2001*, Springer LNCS vol. 2020, 2001, pp. 282–297.

[Pornin] Pornin, T., "Deterministic Usage of DSA and ECDSA Digital Signature Algorithms," IETF work in progress draft-pornin-deterministic-dsa-00, March 2011.

[RFC2154] Murphy, S., M. Badger, and B. Wellington, "OSPF with Digital Signatures," RFC 2154, June 1997. (Experimental)

[RFC2747] Baker, F., B. Lindell, and M. Talwar, "RSVP Cryptographic Authentication," IETF RFC 2747, January 2000.

[RFC3936] Kompella, K., and J. Lang, "Procedures for Modifying the Resource reSerVation Protocol (RSVP)," IETF RFC 3936, October 2004.

[RFC4086] Eastlake, D., 3rd, J. Schiller, and S. Crocker, "Randomness Requirements for Security," IETF RFC 4086, June 2005.

[RFC4949] Shirey, R., "Internet Security Glossary, Version 2," IETF RFC 4949, August 2007.

[TN00] Talwar, V., and K. Nahrstedt, "Securing RSVP For Multimedia Applications," *ACM Multimedia Workshop*, 2000, pp. 153–156.

[Wu99] Wu, T.-L., et al., "Securing QoS: Threats to RSVP Messages and their Countermeasures," *IWQoS'99*, pp.62–64.

[ZSS04] Zhang, F., R. Safavi-Naini, and W. Susilo, "An Efficient Signature Scheme from Bilinear Pairing and its Applications," in: F. Bao et al., eds., *PKC 2004*, Springer LNCS vol. 2947, 2004, pp. 277–290.

Appendix A: Glossary

A thorough glossary of Internet and TCP/IP security terminology can be found in [RFC4949].

Appendix B: OIF Members When the Document Was Approved

Acacia Communications
ADVA Optical Networking
Alcatel-Lucent
Altera
AMCC
Amphenol Corp.
Anritsu
Applied Communication Sciences
AT&T
Avago Technologies Inc.
Broadcom
Brocade
Centellax, Inc.
China Telecom
Ciena Corporation
Cisco Systems
ClariPhy Communications
Cogo Optronics
Comcast
Cortina Systems
CyOptics
Dell, Inc.
Department of Defense
Deutsche Telekom
ECI Telecom Ltd.
Emcore
Emulex
Ericsson
ETRI
EXFO
FCI USA LLC
Fiberhome Technologies Group
Finisar Corporation
France Telecom Group/Orange
Fujitsu
Fundacao.CPqD
Furukawa Electric Japan
GigOptix Inc.
Hewlett Packard
Hitachi
Hittite Microwave Corp
Huawei Technologies
IBM Corporation
Infinera
Inphi
IP Infusion
JDSU
Juniper Networks
KDDI R&D Laboratories
Kotura, Inc.
LeCroy
LSI Corporation
Luxtera
M/A-COM Technology Solutions, Inc.
Marben Products
Maxim Integrated Products
Mayo Clinic
Metaswitch
Mitsubishi Electric Corporation
Molex
MoSys, Inc.
NEC
NeoPhotonics
Nokia Siemens Networks
NTT Corporation
Oclaro
Opnext
PETRA
Picometrix
PMC Sierra
QLogic Corporation
Reflex Photonics
Semtech
SHF Communication Technologies
Sumitomo Electric Industries
Sumitomo Osaka Cement
TE Connectivity
Tektronix
Tellabs
TeraXion
Texas Instruments
Time Warner Cable
TriQuint Semiconductor
u2t Photonics AG
Verizon
Vitesse Semiconductor
Xilinx
Xtera Communications
Yamaichi Electronics Ltd.
ZTE Corporation