Internet Engineering Task Force (IETF)
Request for Comments: 5776
Category: Experimental
ISSN: 2070-1721

V. Roca
A. Francillon
S. Faurite
INRIA

April 2010

Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols

Abstract

This document details the Timed Efficient Stream Loss-Tolerant Authentication (TESLA) packet source authentication and packet integrity verification protocol and its integration within the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) content delivery protocols. This document only considers the authentication/integrity verification of the packets generated by the session's sender. The authentication and integrity verification of the packets sent by receivers, if any, is out of the scope of this document.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5776.

Table of Contents

1. Introduction 5
1.1. Scope of This Document 6
1.2. Conventions Used in This Document 7
1.3. Terminology and Notations 7
1.3.1. Notations and Definitions Related to Cryptographic Functions 7
1.3.2. Notations and Definitions Related to Time 8
2. Using TESLA with ALC and NORM: General Operations 9
2.1. ALC and NORM Specificities That Impact TESLA 9
2.2. Bootstrapping TESLA 10
2.2.1. Bootstrapping TESLA with an Out-Of-Band Mechanism . . 10
2.2.2. Bootstrapping TESLA with an In-Band Mechanism 11
2.3. Setting Up a Secure Time Synchronization 11
2.3.1. Direct Time Synchronization 12
2.3.2. Indirect Time Synchronization 12
2.4. Determining the Delay Bounds 13
2.4.1. Delay Bound Calculation in Direct Time Synchronization Mode 14
2.4.2. Delay Bound Calculation in Indirect Time Synchronization Mode 14
2.5. Cryptographic Parameter Values 15
3. Sender Operations 16
3.1. TESLA Parameters 16
3.1.1. Time Intervals 16
3.1.2. Key Chains 16
3.1.3. Time Interval Schedule 20
3.1.4. Timing Parameters 20
3.2. TESLA Signaling Messages 21
3.2.1. Bootstrap Information 21
3.2.2. Direct Time Synchronization Response 22
3.3. TESLA Authentication Information 22
3.3.1. Authentication Tags 23
3.3.2. Digital Signatures 23
3.3.3. Group MAC Tags 24
3.4. Format of TESLA Messages and Authentication Tags 25
3.4.1. Format of a Bootstrap Information Message 26
3.4.2. Format of a Direct Time Synchronization Response . . . 31
3.4.3. Format of a Standard Authentication Tag 32
3.4.4. Format of an Authentication Tag without Key Disclosure 33
3.4.5. Format of an Authentication Tag with a "New Key Chain" Commitment 34
3.4.6. Format of an Authentication Tag with a "Last Key of Old Chain" Disclosure 35
4. Receiver Operations 36
4.1. Verification of the Authentication Information 36

4.1.1. Processing the Group MAC Tag 36
4.1.2. Processing the Digital Signature 37
4.1.3. Processing the Authentication Tag 37
4.2. Initialization of a Receiver 38
4.2.1. Processing the Bootstrap Information Message 38
4.2.2. Performing Time Synchronization 38
4.3. Authentication of Received Packets 40
4.3.1. Discarding Unnecessary Packets Earlier 43
4.4. Flushing the Non-Authenticated Packets of a Previous Key Chain 43
5. Integration in the ALC and NORM Protocols 44
5.1. Authentication Header Extension Format 44
5.2. Use of Authentication Header Extensions 45
5.2.1. EXT_AUTH Header Extension of Type Bootstrap Information 45
5.2.2. EXT_AUTH Header Extension of Type Authentication Tag 48
5.2.3. EXT_AUTH Header Extension of Type Direct Time Synchronization Request 49
5.2.4. EXT_AUTH Header Extension of Type Direct Time Synchronization Response 49
6. Security Considerations 50
6.1. Dealing with DoS Attacks 50
6.2. Dealing With Replay Attacks 51
6.2.1. Impacts of Replay Attacks on TESLA 51
6.2.2. Impacts of Replay Attacks on NORM 52
6.2.3. Impacts of Replay Attacks on ALC 53
6.3. Security of the Back Channel 53
7. IANA Considerations 54
8. Acknowledgments 55
9. References 55
9.1. Normative References 55
9.2. Informative References 56

1. Introduction

Many applications using multicast and broadcast communications require that each receiver be able to authenticate the source of any packet it receives as well as the integrity of these packets. This is the case with ALC [RFC5775] and NORM [RFC5740], two Content Delivery Protocols (CDPs) designed to transfer objects (e.g., files) reliably between a session's sender and several receivers. The NORM protocol is based on bidirectional transmissions. Each receiver acknowledges data received or, in case of packet erasures, asks for retransmissions. On the opposite, the ALC protocol is based on purely unidirectional transmissions. Reliability is achieved by means of the cyclic transmission of the content within a carousel and/or by the use of proactive Forward Error Correction (FEC) codes. Both protocols have in common the fact that they operate at the application level, on top of an erasure channel (e.g., the Internet) where packets can be lost (erased) during the transmission.

The goal of this document is to counter attacks where an attacker impersonates the ALC or NORM session's sender and injects forged packets to the receivers, thereby corrupting the objects reconstructed by the receivers.

Preventing this attack is much more complex in the case of group communications than it is with unicast communications. Indeed, with unicast communications, a simple solution exists: the sender and the receiver share a secret key to compute a Message Authentication Code (MAC) of all messages exchanged. This is no longer feasible in the case of multicast and broadcast communications since sharing a group key between the sender and all receivers implies that any group member can impersonate the sender and send forged messages to other receivers.

The usual solution to provide the source authentication and message integrity services in the case of multicast and broadcast communications consists of relying on asymmetric cryptography and using digital signatures. Yet, this solution is limited by high computational costs and high transmission overheads. The Timed Efficient Stream Loss-tolerant Authentication (TESLA) protocol is an alternative solution that provides the two required services, while being compatible with high-rate transmissions over lossy channels.

This document explains how to integrate the TESLA source authentication and packet integrity protocol to the ALC and NORM CDP. Any application built on top of ALC and NORM will directly benefit from the services offered by TESLA at the transport layer. In particular, this is the case of File Delivery over Unidirectional Transport (FLUTE).

For more information on the TESLA protocol and its principles, please refer to [RFC4082] and [Perrig04]. For more information on ALC and NORM, please refer to [RFC5775], [RFC5651], and [RFC5740], respectively. For more information on FLUTE, please refer to [RMT-FLUTE].

1.1. Scope of This Document

This specification only considers the authentication and integrity verification of the packets generated by the session's sender. This specification does not consider the packets that may be sent by receivers, for instance, NORM's feedback packets. [RMT-SIMPLE-AUTH] describes several techniques that can be used to that purpose. Since this is usually a low-rate flow (unlike the downstream flow), using computing intensive techniques like digital signatures, possibly combined with a Group MAC scheme, is often acceptable. Finally, Section 5 explains how to use several authentication schemes in a given session thanks to the "ASID" (Authentication Scheme IDentifier) field.

This specification relies on several external mechanisms, for instance:

- to communicate securely the public key or a certificate for the session's sender (Section 2.2.2);
- to communicate securely and confidentially the group key, K_g, used by the Group MAC feature, when applicable (Section 3.3.3). In some situations, this group key will have to be periodically refreshed;
- to perform secure time synchronization in indirect mode (Section 2.3.2) or in direct mode (Section 2.3.1) to carry the request/response messages with ALC, which is purely unidirectional;

These mechanisms are required in order to bootstrap TESLA at a sender and at a receiver and must be deployed in parallel to TESLA. Besides, the randomness of the Primary Key of the key chain (Section 3.1.2) is vital to the security of TESLA. Therefore, the sender needs an appropriate mechanism to generate this random key.

Several technical details of TESLA, like the most appropriate way to alternate between the transmission of a key disclosure and a commitment to a new key chain, or the transmission of a key disclosure and the last key of the previous key chain, or the disclosure of a key and the compact flavor that does not disclose any key, are specific to the target use case (Section 3.1.2). For

instance, it depends on the number of packets sent per time interval, on the desired robustness and the acceptable transmission overhead, which can only be optimized after taking into account the use-case specificities.

1.2. Conventions Used in This Document

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 [RFC2119].

1.3. Terminology and Notations

The following notations and definitions are used throughout this document.

1.3.1. Notations and Definitions Related to Cryptographic Functions

Notations and definitions related to cryptographic functions [RFC4082][RFC4383]:

- PRF is the Pseudo Random Function;
- MAC is the Message Authentication Code;
- HMAC is the keyed-Hash Message Authentication Code;
- F is the one-way function used to create the key chain (Section 3.1.2.1);
- F' is the one-way function used to derive the HMAC keys (Section 3.1.2.1);
- n_p is the length, in bits, of the F function's output. This is therefore the length of the keys in the key chain;
- n_f is the length, in bits, of the F' function's output. This is therefore the length of the HMAC keys;
- n_m is the length, in bits, of the truncated output of the MAC [RFC2104]. Only the n_m most significant bits of the MAC output are kept;
- N is the length of a key chain. There are N+1 keys in a key chain: K_0, K_1, ..., K_N. When several chains are used, all the chains MUST have the same length and keys are numbered consecutively, following the time interval numbering;

- n_c is the number of keys in a key chain. Therefore, n_c = N+1;

- n_tx_lastkey is the number of additional intervals during which the last key of the old key chain SHOULD be sent, after switching to a new key chain and after waiting for the disclosure delay d. These extra transmissions take place after the interval during which the last key is normally disclosed. The n_tx_lastkey value is either 0 (no extra disclosure) or larger. This parameter is sender specific and is not communicated to the receiver;

- n_tx_newkcc is the number of intervals during which the commitment to a new key chain SHOULD be sent, before switching to the new key chain. The n_tx_newkcc value is either 0 (no commitment sent within authentication tags) or larger. This parameter is sender specific and is not communicated to the receiver;

- K_g is a shared group key, communicated to all group members, confidentially, during the TESLA bootstrapping (Section 2.2);

- n_w is the length, in bits, of the truncated output of the MAC of the optional group authentication scheme: only the n_w most significant bits of the MAC output are kept. n_w is typically small, a multiple of 32 bits (e.g., 32 bits).

1.3.2. Notations and Definitions Related to Time

Notations and definitions related to time:

- i is the time interval index. Interval numbering starts at 0 and increases consecutively. Since the interval index is stored as a 32-bit unsigned integer, wrapping to 0 might take place in long sessions.

- t_s is the sender local time value at some absolute time (in NTP timestamp format);

- t_r is the receiver local time value at the same absolute time (in NTP timestamp format);

- T_0 is the start time corresponding to the beginning of the session, i.e., the beginning of time interval 0 (in NTP timestamp format);

- T_int is the interval duration (in milliseconds);

- d is the key disclosure delay (in number of intervals);

- D_t is the upper bound of the lag of the receiver's clock with respect to the clock of the sender;
- S_sr is an estimated bound of the clock drift between the sender and a receiver throughout the duration of the session;
- D^O_t is the upper bound of the lag of the sender's clock with respect to the time reference in indirect time synchronization mode;
- D^R_t is the upper bound of the lag of the receiver's clock with respect to the time reference in indirect time synchronization mode;
- D_err is an upper bound of the time error between all the time references, in indirect time synchronization mode;
- NTP timestamp format consists in a 64-bit unsigned fixed-point number, in seconds relative to 0h on 1 January 1900. The integer part is in the first 32 bits, and the fraction part in the last 32 bits [RFC1305].

2. Using TESLA with ALC and NORM: General Operations

2.1. ALC and NORM Specificities That Impact TESLA

The ALC and NORM protocols have features and requirements that largely impact the way TESLA can be used.

In the case of ALC:

- ALC is massively scalable: nothing in the protocol specification limits the number of receivers that join a session. Therefore, an ALC session potentially includes a huge number (e.g., millions or more) of receivers;
- ALC can work on top of purely unidirectional transport channels: this is one of the assets of ALC, and examples of unidirectional channels include satellite (even if a back channel might exist in some use cases) and broadcasting networks like Digital Video Broadcasting - Handhelds / Satellite services to Handhelds (DVB-H/SH);
- ALC defines an on-demand content delivery model [RFC5775] where receivers can arrive at any time, at their own discretion, download the content and leave the session. Other models (e.g., push or streaming) are also defined;

- ALC sessions are potentially very long: a session can last several days or months during which the content is continuously transmitted within a carousel. The content can be either static (e.g., a software update) or dynamic (e.g., a web site).

Depending on the use case, some of the above features may not apply. For instance, ALC can also be used over a bidirectional channel or with a limited number of receivers.

In the case of NORM:

- NORM has been designed for medium-size sessions: indeed, NORM relies on feedback messages and the sender may collapse if the feedback message rate is too high;

- NORM requires a bidirectional transport channel: the back channel is not necessarily a high-data rate channel since the control traffic sent over it by a single receiver is an order of magnitude lower than the downstream traffic. Networks with an asymmetric connectivity (e.g., a high-rate satellite downlink and a low-rate return channel) are appropriate.

2.2. Bootstrapping TESLA

In order to initialize the TESLA component at a receiver, the sender MUST communicate some key information in a secure way, so that the receiver can check the source of the information and its integrity. Two general methods are possible:

- by using an out-of-band mechanism, or

- by using an in-band mechanism.

The current specification does not recommend any mechanism to bootstrap TESLA. Choosing between an in-band and out-of-band scheme is left to the implementer, depending on the target use case. However, it is RECOMMENDED that TESLA implementations support the use of the in-band mechanism for interoperability purposes.

2.2.1. Bootstrapping TESLA with an Out-Of-Band Mechanism

For instance, [RFC4442] describes the use of the MIKEY (Multimedia Internet Keying) protocol to bootstrap TESLA. As a side effect, MIKEY also provides a loose time synchronization feature from which TESLA can benefit. Other solutions, for instance, based on an extended session description, are possible, on the condition that these solutions provide the required security level.

2.2.2. Bootstrapping TESLA with an In-Band Mechanism

This specification describes an in-band mechanism. In some use cases, it might be desired that bootstrapping take place without requiring the use of an additional external mechanism. For instance, each device may feature a clock with a known time-drift that is negligible in front of the time accuracy required by TESLA, and each device may embed the public key of the sender. It is also possible that the use case does not feature a bidirectional channel that prevents the use of out-of-band protocols like MIKEY. For these two examples, the exchange of a bootstrap information message (described in Section 3.4.1) and the knowledge of a few additional parameters (listed below) are sufficient to bootstrap TESLA at a receiver.

Some parameters cannot be communicated in-band. In particular:

- the sender or group controller MUST either communicate the public key of the sender or a certificate (which also means that a PKI has been set up) to all receivers, so that each receiver be able to verify the signature of the bootstrap message and direct time synchronization response messages (when applicable).

- when time synchronization is performed with NTP/SNTP (Simple Network Time Protocol), the sender or group controller MUST communicate the list of valid NTP/SNTP servers to all the session members (sender included), so that they are all able to synchronize themselves on the same NTP/SNTP servers.

- when the Group MAC feature is used, the sender or group controller MUST communicate the K_g group key to all the session members (sender included). This group key may be periodically refreshed.

The way these parameters are communicated is out of the scope of this document.

2.3. Setting Up a Secure Time Synchronization

The security offered by TESLA heavily relies on time. Therefore, the session's sender and each receiver need to be time synchronized in a secure way. To that purpose, two general methods exist:

- direct time synchronization, and

- indirect time synchronization.

It is also possible that a given session includes receivers that use the direct time synchronization mode while others use the indirect time synchronization mode.

2.3.1. Direct Time Synchronization

When direct time synchronization is used, each receiver asks the sender for a time synchronization. To that purpose, a receiver sends a direct time synchronization request (Section 4.2.2.1). The sender then directly answers each request with a direct time synchronization response (Section 3.4.2), signing this reply. Upon receiving this response, a receiver first verifies the signature, and then calculates an upper bound of the lag of his clock with respect to the clock of the sender, D_t. The details on how to calculate D_t are given in Section 2.4.1.

This synchronization method is both simple and secure. Yet, there are two potential issues:

- a bidirectional channel must exist between the sender and each receiver, and

- the sender may collapse if the incoming request rate is too high.

Relying on direct time synchronization is not expected to be an issue with NORM since (1) bidirectional communications already take place, and (2) NORM scalability is anyway limited. Yet, it can be required that a mechanism, that is out of the scope of this document, be used to spread the transmission of direct time synchronization request messages over time if there is a risk that the sender may collapse.

But direct time synchronization is potentially incompatible with ALC since (1) there might not be a back channel, and (2) there are potentially a huge number of receivers and therefore a risk that the sender will collapse.

2.3.2. Indirect Time Synchronization

When indirect time synchronization is used, the sender and each receiver must synchronize securely via an external time reference. Several possibilities exist:

- sender and receivers can synchronize through an NTPv3 (Network Time Protocol version 3) [RFC1305] hierarchy of servers. The authentication mechanism of NTPv3 MUST be used in order to authenticate each NTP message individually. It prevents, for instance, an attacker from impersonating an NTP server;

- they can synchronize through an NTPv4 (Network Time Protocol version 4) [NTP-NTPv4] hierarchy of servers. The Autokey security protocol of NTPv4 MUST be used in order to authenticate each NTP message individually;

- they can synchronize through an SNTPv4 (Simple Network Time Protocol version 4) [RFC4330] hierarchy of servers. The authentication features of SNTPv4 must then be used. Note that TESLA only needs a loose (but secure) time synchronization, which is in line with the time synchronization service offered by SNTP;

- they can synchronize through a GPS or Galileo (or similar) device that also provides a high precision time reference. Spoofing attacks on the GPS system have recently been reported. Depending on the use case, the security achieved will or will not be acceptable;

- they can synchronize thanks to a dedicated hardware, embedded on each sender and receiver, that provides a clock with a time-drift that is negligible in front of the TESLA time accuracy requirements. This feature enables a device to synchronize its embedded clock with the official time reference from time to time (in an extreme case once, at manufacturing time), and then to remain autonomous for a duration that depends on the known maximum clock drift.

A bidirectional channel is required by the NTP/SNTP schemes. On the opposite, with the GPS/Galileo and high precision clock schemes, no such assumption is made. In situations where ALC is used on purely unidirectional transport channels (Section 2.1), using the NTP/SNTP schemes is not possible. Another aspect is the scalability requirement of ALC, and to a lesser extent of NORM. From this point of view, the above mechanisms usually do not raise any problem, unlike the direct time synchronization schemes. Therefore, using indirect time synchronization can be a good choice. It should be noted that the NTP/SNTP schemes assume that each client trusts the sender and accepts aligning its NTP/SNTP configuration to that of the sender. If this assumption does not hold, the sender SHOULD offer an alternative solution.

The details on how to calculate an upper bound of the lag of a receiver's clock with respect to the clock of the sender, D_t, are given in Section 2.4.2.

2.4. Determining the Delay Bounds

Let us assume that a secure time synchronization has been set up. This section explains how to define the various timing parameters that are used during the authentication of received packets.

2.4.1. Delay Bound Calculation in Direct Time Synchronization Mode

In direct time synchronization mode, synchronization between a receiver and the sender follows the following protocol [RFC4082]:

- The receiver sends a direct time synchronization request message to the sender, that includes t_r, the receiver local time at the moment of sending (Section 4.2.2.1).

- Upon receipt of this message, the sender records its local time, t_s, and sends to the receiver a direct time synchronization response that includes t_r (taken from the request) and t_s, signing this reply (Section 3.4.2).

- Upon receiving this response, the receiver first verifies that he actually sent a request with t_r and then checks the signature. Then he calculates D_t = t_s - t_r + S_sr, where S_sr is an estimated bound of the clock drift between the sender and the receiver throughout the duration of the session. This document does not specify how S_sr is estimated.

After this initial synchronization, at any point throughout the session, the receiver knows that: T_s < T_r + D_t, where T_s is the current time at the sender and T_r is the current time at the receiver.

2.4.2. Delay Bound Calculation in Indirect Time Synchronization Mode

In indirect time synchronization, the sender and the receivers must synchronize indirectly using one or several time references.

2.4.2.1. Single Time Reference

Let us assume that there is a single time reference.

1. The sender calculates D^O_t, the upper bound of the lag of the sender's clock with respect to the time reference. This D^O_t value is then communicated to the receivers (Section 3.2.1).

2. Similarly, a receiver R calculates D^R_t, the upper bound of the lag of the receiver's clock with respect to the time reference.

3. Then, for receiver R, the overall upper bound of the lag of the receiver's clock with respect to the clock of the sender, D_t, is the sum: D_t = D^O_t + D^R_t.

The D^O_t and D^R_t calculation depends on the time synchronization mechanism used (Section 2.3.2). In some cases, the synchronization scheme specifications provide these values. In other cases, these parameters can be calculated by means of a scheme similar to the one specified in Section 2.4.1, for instance, when synchronization is achieved via a group controller [RFC4082].

2.4.2.2. Multiple Time References

Let us now assume that there are several time references (e.g., several NTP/SNTP servers). The sender and receivers first synchronize with the various time references, independently. It results in D^O_t and D^R_t. Let D_err be an upper bound of the time error between all of the time references. Then, the overall value of D_t within receiver R is set to the sum: D_t = D^O_t + D^R_t + D_err.

In some cases, the D_t value is part of the time synchronization scheme specifications. For instance, NTPv3 [RFC1305] defines algorithms that are "capable of accuracies in the order of a millisecond, even after extended periods when synchronization to primary reference sources has been lost". In practice, depending on the NTP server stratum, the accuracy might be a little bit worse. In that case, D_t = security_factor * (1ms + 1ms), where the security_factor is meant to compensate several sources of inaccuracy in NTP. The choice of the security_factor value is left to the implementer, depending on the target use case.

2.5. Cryptographic Parameter Values

The F (resp. F') function output length is given by the n_p (resp. n_f) parameter. The n_p and n_f values depend on the PRF function chosen, as specified below:

PRF name	n_p and n_f
HMAC-SHA-1	160 bits (20 bytes)
HMAC-SHA-224	224 bits (28 bytes)
HMAC-SHA-256 (default)	256 bits (32 bytes)
HMAC-SHA-384	384 bits (48 bytes)
HMAC-SHA-512	512 bits (64 bytes)

The computing of regular MAC (resp. Group MAC) makes use of the n_m (resp. n_w) parameter, i.e., the length of the truncated output of the function. The n_m and n_w values depend on the MAC function chosen, as specified below:

MAC name	n_m (regular MAC)	n_w (Group MAC)
HMAC-SHA-1	80 bits (10 bytes)	32 bits (4 bytes)
HMAC-SHA-224	112 bits (14 bytes)	32 bits (4 bytes)
HMAC-SHA-256 (default)	128 bits (16 bytes)	32 bits (4 bytes)
HMAC-SHA-384	192 bits (24 bytes)	32 bits (4 bytes)
HMAC-SHA-512	256 bits (32 bytes)	32 bits (4 bytes)

3. Sender Operations

This section describes the TESLA operations at a sender. For more information on the TESLA protocol and its principles, please refer to [RFC4082][Perrig04].

3.1. TESLA Parameters

3.1.1. Time Intervals

The sender divides the time into uniform intervals of duration T_int. Time interval numbering starts at 0 and is incremented consecutively. The interval index MUST be stored in an unsigned 32-bit integer so that wrapping to 0 takes place only after 2^^32 intervals. For instance, if T_int is equal to 0.5 seconds, then wrapping takes place after approximately 68 years.

3.1.2. Key Chains

3.1.2.1. Principles

The sender computes a one-way key chain of n_c = N+1 keys, and assigns one key from the chain to each interval, consecutively but in reverse order. Key numbering starts at 0 and is incremented consecutively, following the time interval numbering: K_0, K_1, ..., K_N.

In order to compute this chain, the sender must first select a Primary Key, K_N, and a PRF function, f (Section 7, TESLA-PRF). The randomness of the Primary Key, K_N, is vital to the security and no one should be able to guess it.

The function F is a one-way function that is defined as: F(k) = f_k(0), where f_k(0) is the result of the application of the PRF f to k and 0. When f is an HMAC (Section 7), k is used as the key, and 0 as the message, using the algorithm described in [RFC2104].

Similarly, the function F' is a one-way function that is defined as: F'(k) = f_k(1), where f_k(1) is the result of the application of the same PRF f to k and 1.

The sender then computes all the keys of the chain, recursively, starting with K_N, using: K_{i-1} = F(K_i). Therefore, K_i = F^{N-i}(K_N), where F^i(x) is the execution of function F with the argument x, i times. The receiver can then compute any value in the key chain from K_N, even if it does not have intermediate values [RFC4082]. The key for MAC calculation can then be derived from the corresponding K_i key by K'_i = F'(K_i).

The key chain has a finite length, N, which corresponds to a maximum time duration of (N + 1) * T_int. The content delivery session has a duration T_delivery, which may either be known in advance, or not. A first solution consists in having a single key chain of an appropriate length, so that the content delivery session finishes before the end of the key chain, i.e., T_delivery <= (N + 1) * T_int. But the longer the key chain, the higher the memory and computation required to cope with it. Another solution consists in switching to a new key chain, of the same length, when necessary [Perrig04].

3.1.2.2. Using Multiple Key Chains

When several key chains are needed, all of them MUST be of the same length. Switching from the current key chain to the next one requires that a commitment to the new key chain be communicated in a secure way to the receiver. This can be done by using either an out-of-band mechanism or an in-band mechanism. This document only specifies the in-band mechanism.

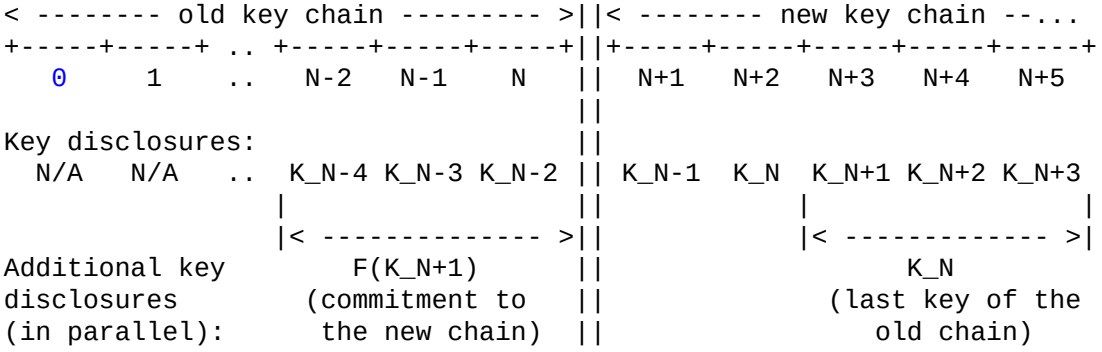

Figure 1: Switching to the Second Key Chain with the In-Band Mechanism, Assuming That d=2, n_tx_newkcc=3, n_tx_lastkey=3

Figure 1 illustrates the switch to the new key chain, using the in-band mechanism. Let us say that the old key chain stops at K_N and the new key chain starts at K_{N+1} (i.e., F(K_{N+1}) and K_N are two different keys). Then, the sender includes the commitment F(K_{N+1}) to the new key chain into packets authenticated with the old key chain (see Section 3.4.5). This commitment SHOULD be sent during n_tx_newkcc time intervals before the end of the old key chain. Since several packets are usually sent during an interval, the sender SHOULD alternate between sending a disclosed key of the old key chain and the commitment to the new key chain. The details of how to alternate between the disclosure and commitment are out of the scope of this document.

The receiver will keep the commitment until the key K_{N+1} is disclosed, at interval N+1+d. Then, the receiver will be able to test the validity of that key by computing F(K_{N+1}) and comparing it to the commitment.

When the key chain is changed, it becomes impossible to recover a previous key from the old key chain. This is a problem if the receiver lost the packets disclosing the last key of the old key chain. A solution consists in re-sending the last key, K_N, of the old key chain (see Section 3.4.6). This SHOULD be done during n_tx_lastkey additional time intervals after the end of the time interval where K_N is disclosed. Since several packets are usually sent during an interval, the sender SHOULD alternate between sending a disclosed key of the new key chain, and the last key of the old key chain. The details of how to alternate between the two disclosures are out of the scope of this document.

In some cases, a receiver having experienced a very long disconnection might have lost the commitment of the new chain. Therefore, this receiver will not be able to authenticate any packet related to the new chain or any of the following ones. The only solution for this receiver to catch up consists in receiving an additional bootstrap information message. This can happen by waiting for the next periodic transmission (if sent in-band) or through an external mechanism (Section 3.2.1).

3.1.2.3. Values of the n_tx_lastkey and n_tx_newkcc Parameters

When several key chains and the in-band commitment mechanism are used, a sender MUST initialize the n_tx_lastkey and n_tx_newkcc parameters in such a way that no overlapping occurs. In other words, once a sender starts transmitting commitments for a new key chain, he MUST NOT send a disclosure for the last key of the old key chain any more. Therefore, the following property MUST be verified:

d + n_tx_lastkey + n_tx_newkcc <= N + 1

It is RECOMMENDED, for robustness purposes, that, once n_tx_lastkey has been chosen, then:

n_tx_newkcc = N + 1 - n_tx_lastkey - d

In other words, the sender starts transmitting a commitment to the following key chain immediately after having sent all the disclosures of the last key of the previous key chain. Doing so increases the probability that a receiver gets a commitment for the following key chain.

In any case, these two parameters are sender specific and need not be transmitted to the receivers. Of course, as explained above, the sender alternates between the disclosure of a key of the current key chain and the commitment to the new key chain (or the last key of the old key chain).

3.1.2.4. The Particular Case of the Session Start

Since a key cannot be disclosed before the disclosure delay, d, no key will be disclosed during the first d time intervals (intervals 0 and 1 in Figure 1) of the session. To that purpose, the sender uses the Authentication Tag without Key Disclosure, Section 3.4.4. The following key chains, if any, are not concerned since they will disclose the last d keys of the previous chain.

3.1.2.5. Managing Silent Periods

An ALC or NORM sender may stop transmitting packets for some time. For instance, it can be the end of the session and all packets have already been sent, or the use case may consist in a succession of busy periods (when fresh objects are available) followed by silent periods. In any case, this is an issue since the authentication of the packets sent during the last d intervals requires that the associated keys be disclosed, which will take place during d additional time intervals.

To solve this problem, it is recommended that the sender transmit empty packets (i.e., without payload) containing the TESLA EXT_AUTH Header Extension along with a Standard Authentication Tag during at least d time intervals after the end of the regular ALC or NORM packet transmissions. The number of such packets and the duration during which they are sent must be sufficient for all receivers to receive, with a high probability, at least one packet disclosing the last useful key (i.e., the key used for the last non-empty packet sent).

3.1.3. Time Interval Schedule

The sender must determine the following parameters:

- T_0, the start time corresponding to the beginning of the session, i.e., the beginning of time interval 0 (in NTP timestamp format);
- T_int, the interval duration (in milliseconds), usually ranging from 100 milliseconds to 1 second;
- d, the key disclosure delay (in number of intervals). It is the time to wait before disclosing a key;
- N, the length of a key chain.

The correct choice of T_int, d, and N is crucial for the efficiency of the scheme. For instance, a T_int * d product that is too long will cause excessive delay in the authentication process. A T_int * d product that is too short prevents many receivers from verifying packets. An N * T_int product that is too small will cause the sender to switch too often to new key chains. An N that is too long with respect to the expected session duration (if known) will require the sender to compute too many useless keys. Sections 3.2 and 3.6 of [RFC4082] give general guidelines for initializing these parameters.

The T_0, T_int, d, and N parameters MUST NOT be changed during the lifetime of the session. This restriction is meant to prevent introducing vulnerabilities. For instance, if a sender was authorized to change the key disclosure schedule, a receiver that did not receive the change notification would still believe in the old key disclosure schedule, thereby creating vulnerabilities [RFC4082].

3.1.4. Timing Parameters

In indirect time synchronization mode, the sender must determine the following parameter:

- D^O_t, the upper bound of the lag of the sender's clock with respect to the time reference.

The D^O_t parameter MUST NOT be changed during the lifetime of the session.

3.2. TESLA Signaling Messages

At a sender, TESLA produces two types of signaling information:

- The bootstrap information: it can be either sent out-of-band or in-band. In the latter case, a digitally signed packet contains all the information required to bootstrap TESLA at a receiver;
- The direct time synchronization response, which enables a receiver to finish a direct time synchronization.

3.2.1. Bootstrap Information

In order to initialize the TESLA component at a receiver, the sender must communicate some key information in a secure way. This information can be sent in-band or out-of-band, as discussed in Section 2.2. In this section, we only consider the in-band scheme.

The TESLA bootstrap information message MUST be digitally signed (Section 3.3.2). The goal is to enable a receiver to check the packet source and packet integrity. Then, the bootstrap information can be:

- unicast to a receiver during a direct time synchronization request/response exchange;
- broadcast to all receivers. This is typically the case in indirect time synchronization mode. It can also be used in direct time synchronization mode, for instance, when a large number of clients arrive at the same time, in which case it is more efficient to answer globally.

Let us consider situations where the bootstrap information is broadcast. This message should be broadcast at the beginning of the session, before data packets are actually sent. This is particularly important with ALC or NORM sessions in "push" mode, when all clients join the session in advance. For improved reliability, bootstrap information might be sent a certain number of times.

A periodic broadcast of the bootstrap information message could also be useful when:

- the ALC session uses an "on-demand" mode, clients arriving at their own discretion;

- some clients experience an intermittent connectivity. This is particularly important when several key chains are used in an ALC or NORM session, since there is a risk that a receiver loses all the commitments to the new key chain.

A balance must be found between the signaling overhead and the maximum initial waiting time at the receiver before starting the delayed authentication process. A period of a few seconds for the transmission of this bootstrap information is often a reasonable value.

3.2.2. Direct Time Synchronization Response

In direct time synchronization, upon receipt of a synchronization request, the sender records its local time, t_s, and sends a response message that contains both t_r and t_s (Section 2.4.1). This message is unicast to the receiver. This direct time synchronization response message MUST be digitally signed in order to enable a receiver to check the packet source and packet integrity (Section 3.3.2). The receiver MUST also be able to associate this response and his request, which is the reason why t_r is included in the response message.

3.3. TESLA Authentication Information

At a sender, TESLA produces three types of security tags:

- an authentication tag, in case of data packets, and which contains the MAC of the packet;

- a digital signature, in case of one of the two TESLA signaling packets, namely a bootstrap information message or a direct time synchronization response; and

- an optional group authentication tag, that can be added to all the packets to mitigate attacks coming from outside of the group.

Because of interdependencies, their computation MUST follow a strict order:

- first of all, compute the authentication tag (with data packet) or the digital signature (with signaling packet);

- finally, compute the Group Mac.

3.3.1. Authentication Tags

All the data packets sent MUST have an authentication tag containing:

- the interval index, i, which is also the index of the key used for computing the MAC of this packet;
- the MAC of the message: MAC(K'_i, M), where K'_i=F'(K_i);
- either a disclosed key (which belongs to the current key chain or the previous key chain), or a commitment to a new key chain, or no key at all.

The computation of MAC(K'_i, M) MUST include the ALC or NORM header (with the various header extensions) and the payload (when applicable). The UDP/IP headers MUST NOT be included. During this computation, the "MAC(K'_i, M)" field of the authentication tag MUST be set to 0.

3.3.2. Digital Signatures

The bootstrap information message (with the in-band bootstrap scheme) and direct time synchronization response message (with the indirect time synchronization scheme) both need to be signed by the sender. These two messages contain a "Signature" field to hold the digital signature. The bootstrap information message also contains the "Signature Encoding Algorithm", the "Signature Cryptographic Function", and the "Signature Length" fields that enable a receiver to process the "Signature" field. Note that there are no such "Signature Encoding Algorithm", "Signature Cryptographic Function", and "Signature Length" fields in the case of a direct time synchronization response message since it is assumed that these parameters are already known (i.e., the receiver either received a bootstrap information message before or these values have been communicated out-of-band).

Several "Signature Encoding Algorithms" can be used, including RSASSA-PKCS1-v1_5, the default, and RSASSA-PSS (Section 7). With these encodings, SHA-256 is the default "Signature Cryptographic Function".

The computation of the signature MUST include the ALC or NORM header (with the various header extensions) and the payload when applicable. The UDP/IP headers MUST NOT be included. During this computation, the "Signature" field MUST be set to 0 as well as the optional Group MAC, when present, since this Group MAC is calculated later.

More specifically, from [RFC4359]: Digital signature generation is performed as described in [RFC3447], Section 8.2.1 for RSASSA-PKCS1-v1_5 and Section 8.1.1 for RSASSA-PSS. The authenticated portion of the packet is used as the message M, which is passed to the signature generation function. The signer's RSA private key is passed as K. In summary, (when SHA-256 is used), the signature generation process computes a SHA-256 hash of the authenticated packet bytes, signs the SHA-256 hash using the private key, and encodes the result with the specified RSA encoding type. This process results in a value S, which is the digital signature to be included in the packet.

With RSASSA-PKCS1-v1_5 and RSASSA-PSS signatures, the size of the signature is equal to the "RSA modulus", unless the "RSA modulus" is not a multiple of 8 bits. In that case, the signature MUST be prepended with between 1 and 7 bits set to zero such that the signature is a multiple of 8 bits [RFC4359]. The key size, which in practice is also equal to the "RSA modulus", has major security implications. [RFC4359] explains how to choose this value depending on the maximum expected lifetime of the session. This choice is out of the scope of this document.

3.3.3. Group MAC Tags

An optional Group MAC can be used to mitigate Denial-of-Service (DoS) attacks coming from attackers that are not group members [RFC4082]. This feature assumes that a group key, K_g, is shared by the sender and all receivers. When the attacker is not a group member, the benefits of adding a Group MAC to every packet sent are threefold:

- a receiver can immediately drop faked packets, without having to wait for the disclosure delay, d;
- a sender can immediately drop faked direct time synchronization requests, and avoid checking the digital signature, a computation intensive task;
- a receiver can immediately drop faked direct time synchronization response and bootstrap messages, without having to verify the digital signature, a computation-intensive task.

The computation of the Group MAC, MAC(K_g, M), MUST include the ALC or NORM header (with the various header extensions) and the payload when applicable. The UDP/IP headers MUST NOT be included. During this computation, the "Group MAC" field MUST be set to 0. However, the digital signature (e.g., of a bootstrap message) and the "MAC" fields (e.g., of an authentication tag), when present, MUST have been

calculated since they are included in the Group MAC calculation itself. Then, the sender truncates the MAC output to keep the n_w most significant bits and stores the result in the "Group MAC" field.

This scheme features a few limits:

- it is of no help if a group member (who knows K_g) impersonates the sender and sends forged messages to other receivers;
- it requires an additional MAC computing for each packet, both at the sender and receiver sides;
- it increases the size of the TESLA authentication headers. In order to limit this problem, the length of the truncated output of the MAC, n_w, SHOULD be kept small (e.g., 32 bits) (see [RFC3711], Section 9.5). As a side effect, the authentication service is significantly weakened: the probability of any forged packet being successfully authenticated becomes one in 2^32. Since the Group MAC check is only a pre-check that must be followed by the standard TESLA authentication check, this is not considered to be an issue.

For a given use case, the benefits brought by the Group MAC must be balanced against these limitations.

Note that the Group MAC function can be different from the TESLA MAC function (e.g., it can use a weaker but faster MAC function). Note also that the mechanism by which the group key, K_g, is communicated to all group members, and perhaps periodically updated, is out of the scope of this document.

3.4. Format of TESLA Messages and Authentication Tags

This section specifies the format of the various kinds of TESLA messages and authentication tags sent by the session's sender. Because these TESLA messages are carried as EXT_AUTH Header Extensions of the ALC or NORM packets (Section 5), the following formats do not start on 32-bit word boundaries.

3.4.1. Format of a Bootstrap Information Message

When bootstrap information is sent in-band, the following message is used:

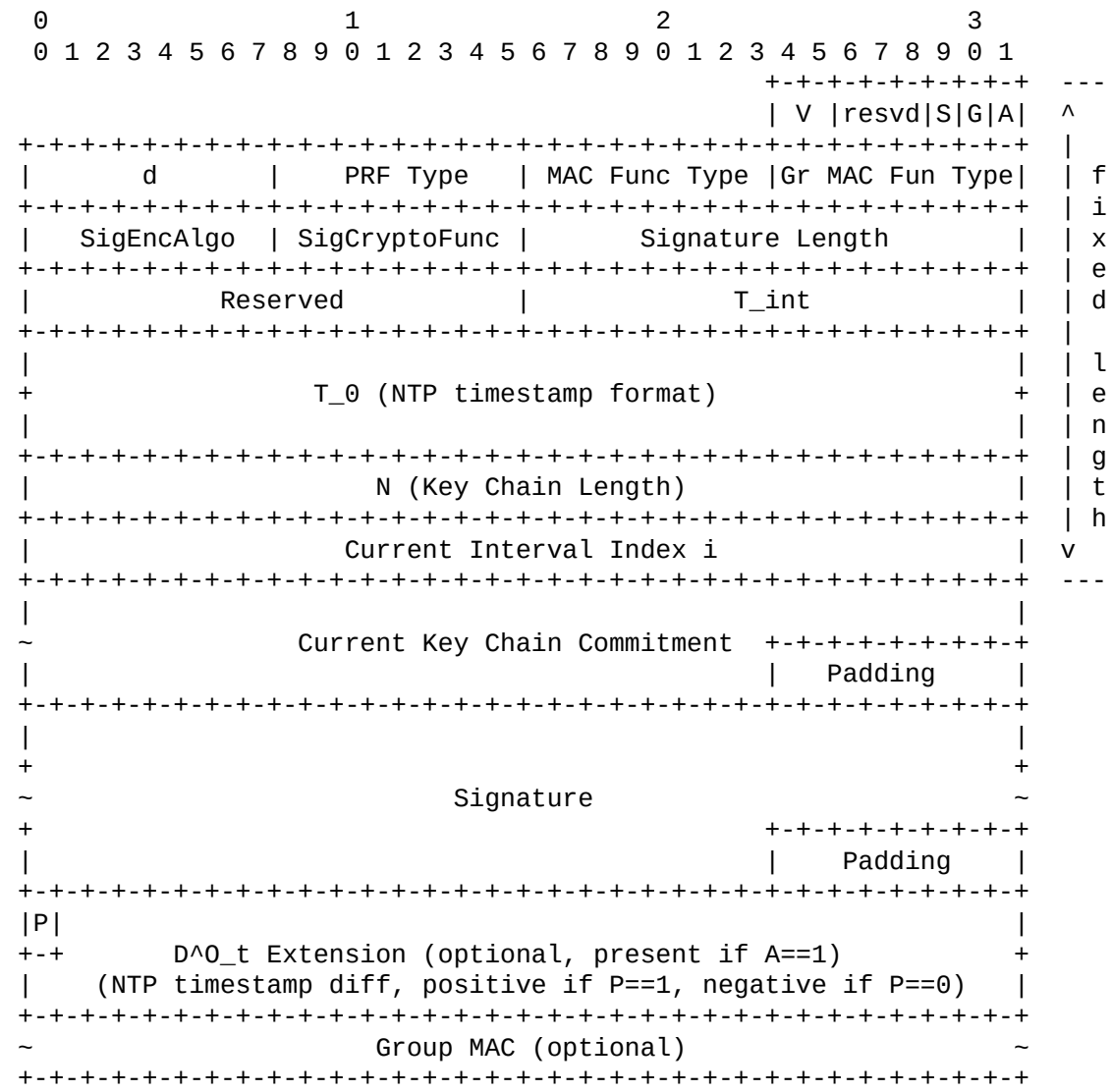

Figure 2: Bootstrap Information Format

The format of the bootstrap information is depicted in Figure 2. The fields are:

"V" (Version) field (2 bits):

The "V" field contains the version number of the protocol. For this specification, the value of 0 MUST be used.

"Reserved" field (3 bits):

This is a reserved field that MUST be set to zero in this specification.

"S" (Single Key Chain) flag (1 bit):

The "S" flag indicates whether this TESLA session is restricted to a single key chain (S==1) or relies on one or multiple key chains (S==0).

"G" (Group MAC Present) flag (1 bit):

The "G" flag indicates whether the Group MAC feature is used (G==1) or not (G==0). When it is used, a "Group MAC" field is added to all the packets containing a TESLA EXT_AUTH Header Extension (including this bootstrap message).

"A" flag (1 bit):

The "A" flag indicates whether the "P" flag and "D^O_t" fields are present (A==1) or not (A==0). In indirect time synchronization mode, A MUST be equal to 1 since these fields are needed.

"d" field (8 bits):

"d" is an unsigned integer that defines the key disclosure delay (in number of intervals). d MUST be greater than or equal to 2.

"PRF Type" field (8 bits):

The "PRF Type" is the reference number of the f function used to derive the F (for key chain) and F' (for MAC keys) functions (Section 7).

"MAC Function Type" field (8 bits):

The "MAC Function Type" is the reference number of the function used to compute the MAC of the packets (Section 7).

"Group MAC Function Type" field (8 bits):

When G==1, this field contains the reference number of the cryptographic MAC function used to compute the Group MAC (Section 7). When G==0, this field MUST be set to zero.

"Signature Encoding Algorithm" field (8 bits):

The "Signature Encoding Algorithm" is the reference number (Section 7) of the digital signature used to authenticate this bootstrap information and included in the "Signature" field.

"Signature Cryptographic Function" field (8 bits):

The "Signature Cryptographic Function" is the reference number (Section 7) of the cryptographic function used within the digital signature.

"Signature Length" field (16 bits):

The "Signature Length" is an unsigned integer that indicates the "Signature" field size in bytes in the "Signature Extension" field. This is also the signature key length, since both parameters are equal.

"Reserved" fields (16 bits):

This is a reserved field that MUST be set to zero in this specification.

"T_int" field (16 bits):

"T_int" is an unsigned 16-bit integer that defines the interval duration (in milliseconds).

"T_0" field (64 bits):

"T_0" is a timestamp in NTP timestamp format that indicates the beginning of the session, i.e., the beginning of time interval 0.

"N" field (32 bits):

"N" is an unsigned integer that indicates the key chain length. There are N + 1 keys per chain.

"i" (Interval Index of K_i) field (32 bits):

"i" is an unsigned integer that indicates the current interval index when this bootstrap information message is sent.

"Current Key Chain Commitment" field (variable size, padded if necessary for 32-bit word alignment):

"Key Chain Commitment" is the commitment to the current key chain, i.e., the key chain corresponding to interval i. For instance, with the first key chain, this commitment is equal to F(K_0), with the second key chain, this commitment is equal to F(K_{N+1}), etc.). If need be, this field is padded (with 0) up to a multiple of 32 bits.

"Signature" field (variable size, padded if necessary for 32-bit word alignment):

The "Signature" field is mandatory. It contains a digital signature of this message, as specified by the encoding algorithm, cryptographic function, and key length parameters. If the signature length is not a multiple of 32 bits, this field is padded with 0.

"P" flag (optional, 1 bit if present):

The "P" flag is optional and only present if the "A" flag is equal to 1. It is only used in indirect time synchronization mode. This flag indicates whether the D^O_t NTP timestamp difference is positive (P==1) or negative (P==0).

"D^O_t" field (optional, 63 bits if present):

The "D^O_t" field is optional and only present if the "A" flag is equal to 1. It is only used in indirect time synchronization mode. It is the upper bound of the lag of the sender’s clock with respect to the time reference. When several time references are specified (e.g., several NTP servers), then D^O_t is the maximum upper bound of the lag with each time reference. D^O_t is composed of two unsigned integers, as with NTP timestamps: the first 31 bits give the time difference in seconds and the remaining 32 bits give the sub-second time difference.

"Group MAC" field (optional, variable length, multiple of 32 bits):

This field contains the Group MAC, calculated with the group key, K_g, shared by all group members. The field length, in bits, is given by n_w, which is known once the Group MAC function type is known (Section 7).

Note that the first byte and the following seven 32-bit words are mandatory fixed-length fields. The "Current Key Chain Commitment" and "Signature" fields are mandatory but variable-length fields. The remaining "D^O_t" and "Group MAC" fields are optional.

In order to prevent attacks, some parameters MUST NOT be changed during the lifetime of the session (Sections 3.1.3 and 3.1.4). The following table summarizes the parameter's status:

Parameter	Status
V	set to 0 in this specification
S	static (during whole session)
G	static (during whole session)
A	static (during whole session)
T_O	static (during whole session)
T_int	static (during whole session)
d	static (during whole session)
N	static (during whole session)
D^O_t (if present)	static (during whole session)
PRF Type	static (during whole session)
MAC Function Type	static (during whole session)
Signature Encoding Algorithm	static (during whole session)
Signature Crypto. Function	static (during whole session)
Signature Length	static (during whole session)
Group MAC Func. Type	static (during whole session)
i	dynamic (related to current key chain)
K_i	dynamic (related to current key chain)
signature	dynamic, packet dependent
Group MAC (if present)	dynamic, packet dependent

3.4.2. Format of a Direct Time Synchronization Response

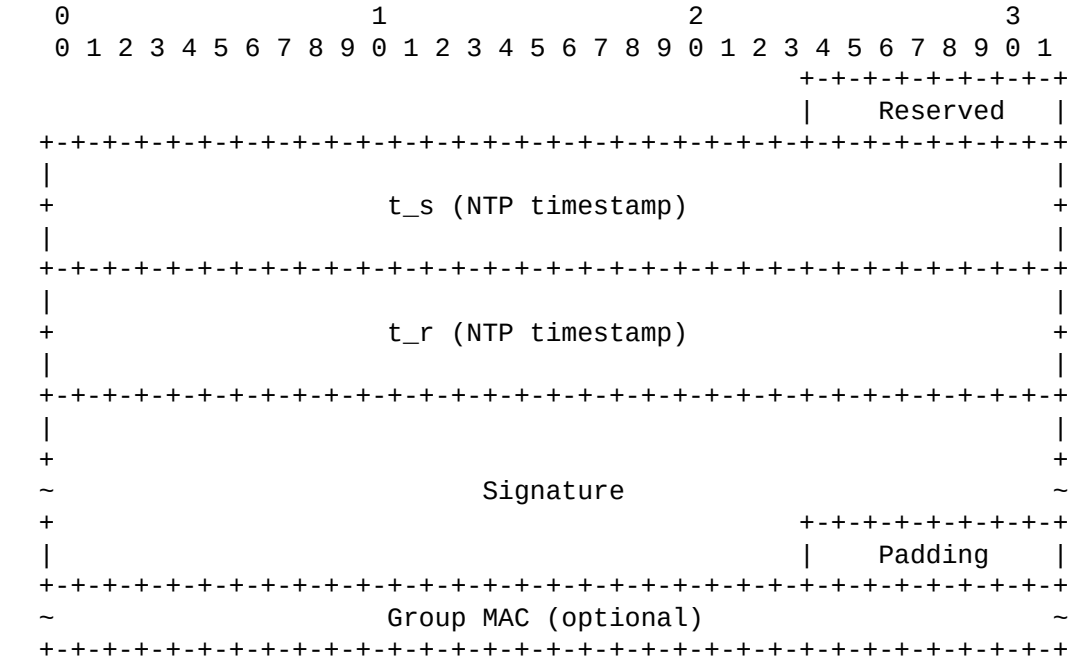

Figure 3: Format of a Direct Time Synchronization Response

The response to a direct time synchronization request contains the following information:

"Reserved" field (8 bits):

This is a reserved field that MUST be set to zero in this specification.

"t_s" (NTP timestamp, 64 bits):

"t_s" is a timestamp in NTP timestamp format that corresponds to the sender local time value when receiving the direct time synchronization request message.

"t_r" (NTP timestamp, 64 bits):

"t_r" is a timestamp in NTP timestamp format that contains the receiver local time value received in the direct time synchronization request message.

"Signature" field (variable size, padded if necessary for 32-bit word alignment):

The "Signature" field is mandatory. It contains a digital signature of this message, as specified by the encoding algorithm, cryptographic function, and key length parameters communicated in the bootstrap information message (if applicable) or out-of-band. If the signature length is not a multiple of 32 bits, this field is padded with 0.

"Group MAC" field (optional, variable length, multiple of 32 bits):

This field contains the Group MAC, calculated with the group key, K_g, shared by all group members. The field length, in bits, is given by n_w, which is known once the Group MAC function type is known (Section 7).

3.4.3. Format of a Standard Authentication Tag

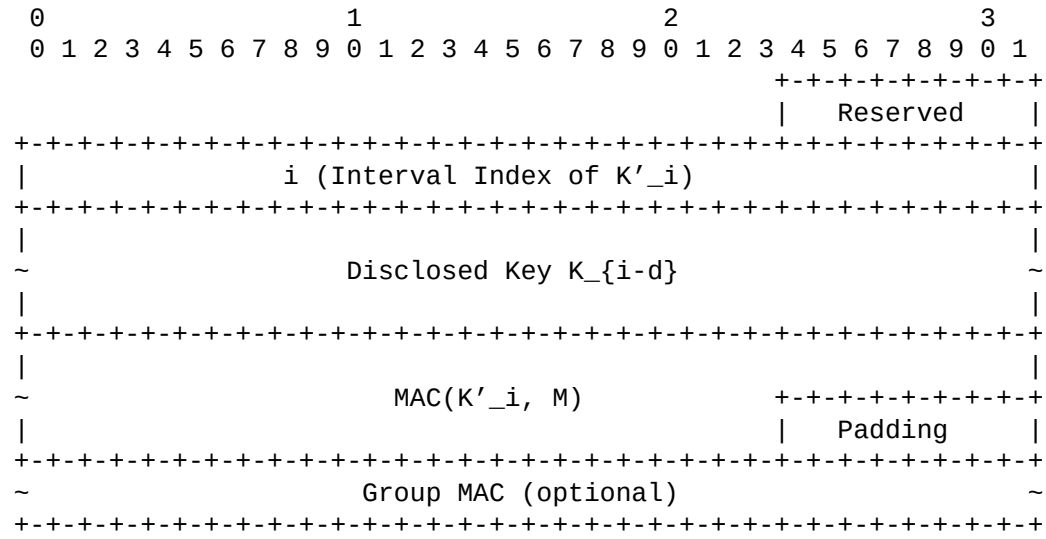

Figure 4: Format of the Standard Authentication Tag

A Standard Authentication Tag is composed of the following fields:

"Reserved" field (8 bits):

The "Reserved" field is not used in the current specification and MUST be set to zero by the sender.

"i" (Interval Index) field (32 bits):

"i" is the interval index associated with the key (K'_i) used to compute the MAC of this packet.

"Disclosed Key" (variable size, non padded):

The "Disclosed Key" is the key used for interval i-d: K_{i-d}. There is no padding between the "Disclosed Key" and "MAC(K'_i, M)" fields, and the latter MAY not start on a 32-bit boundary, depending on the n_p parameter.

"MAC(K'_i, M)" (variable size, padded if necessary for 32-bit word alignment):

"MAC(K'_i, M)" is the truncated message authentication code of the current packet. Only the n_m most significant bits of the MAC output are kept [RFC2104].

"Group MAC" field (optional, variable length, multiple of 32 bits):

This field contains the Group MAC, calculated with a group key, K_g, shared by all group members. The field length is given by n_w, in bits.

Note that because a key cannot be disclosed before the disclosure delay, d, the sender MUST NOT use this tag during the first d intervals of the session: {0 .. d-1} (inclusive). Instead, the sender MUST use an Authentication Tag without Key Disclosure.

3.4.4. Format of an Authentication Tag without Key Disclosure

The Authentication Tag without Key Disclosure is meant to be used in situations where a high number of packets are sent in a given time interval. In such a case, it can be advantageous to disclose the K_{i-d} key only in a subset of the packets sent, using a Standard Authentication Tag, and to use the shortened version that does not disclose the K_{i-d} key in the remaining packets. It is left to the implementer to decide how many packets should disclose the K_{i-d} key. This Authentication Tag without Key Disclosure MUST also be used during the first d intervals: {0 .. d-1} (inclusive).

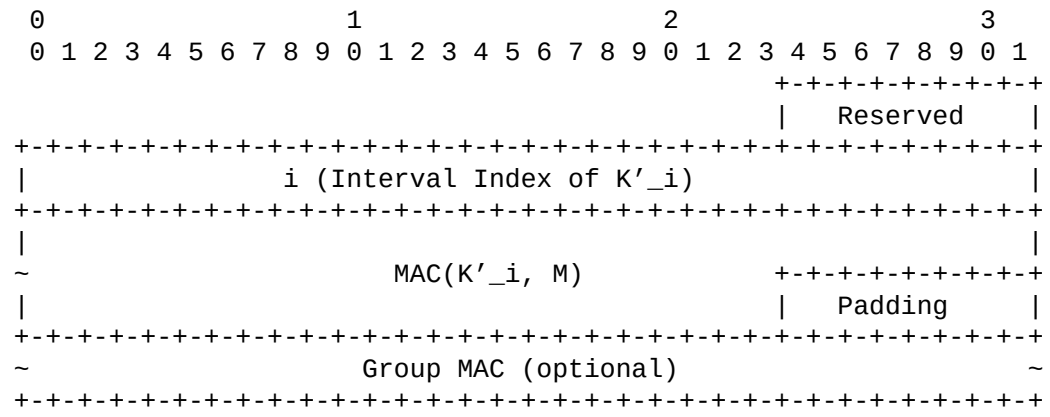

Figure 5: Format of the Authentication Tag without Key Disclosure

3.4.5. Format of an Authentication Tag with a "New Key Chain" Commitment

During the last n_tx_newkcc intervals of the current key chain, the sender SHOULD send commitments to the next key chain. This is done by replacing the disclosed key of the Authentication Tag with a New Key Chain Commitment, F(K_{N+1}) (or F(K_{2N+2}) in case of a switch between the second and third key chains, etc.) Figure 6 shows the corresponding format.

Note that since there is no padding between the "F(K_{N+1})" and "MAC(K'_i, M)" fields, the latter MAY not start on a 32-bit boundary, depending on the n_p parameter.

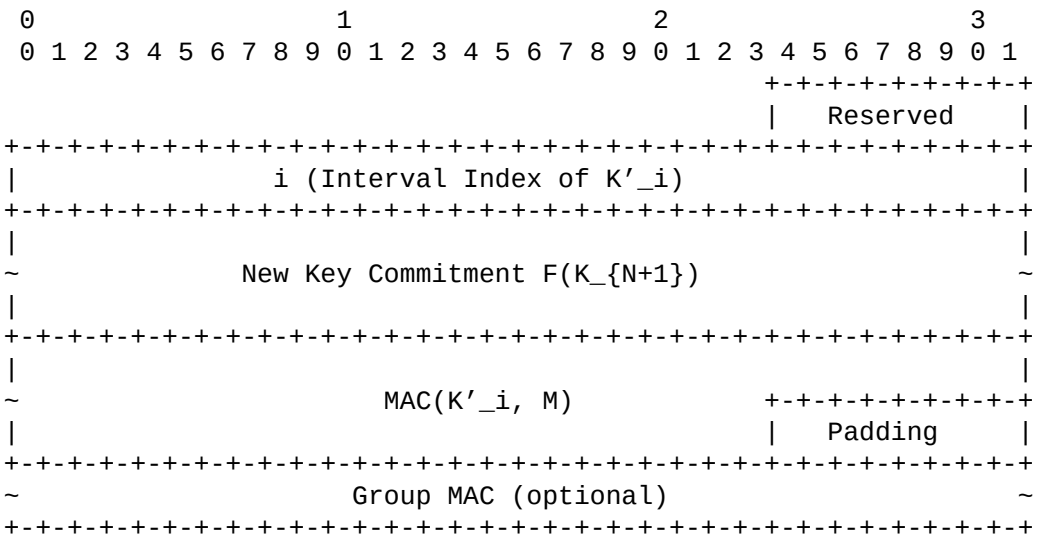

Figure 6: Format of the Authentication Tag
with a New Key Chain Commitment

3.4.6. Format of an Authentication Tag with a "Last Key of Old Chain" Disclosure

During the first n_tx_lastkey intervals of the new key chain after the disclosing interval, d, the sender SHOULD disclose the last key of the old key chain. This is done by replacing the disclosed key of the Authentication Tag with the Last Key of the Old Chain, K_N (or K_{2N+1} in case of a switch between the second and third key chains, etc.). Figure 7 shows the corresponding format.

Note that since there is no padding between the "K_N" and "MAC(K'_i, M)" fields, the latter MAY not start on a 32-bit boundary, depending on the n_p parameter.

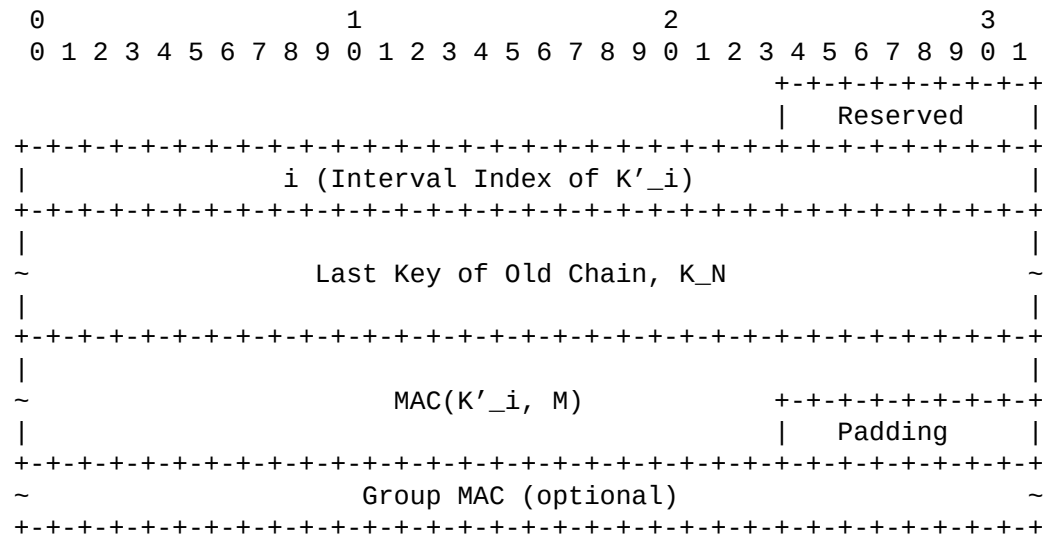

Figure 7: Format of the Authentication Tag
with an Old Chain Last Key Disclosure

4. Receiver Operations

This section describes the TESLA operations at a receiver.

4.1. Verification of the Authentication Information

This section details the computation steps required to verify each of the three possible authentication information of an incoming packet. The verification MUST follow a strict order:

- first of all, if the Group MAC is present and if the session uses this feature (e.g., if the G bit is set in the bootstrap information message), then verify the Group MAC. A packet that does not contain a Group MAC tag, whereas the session uses this feature, MUST be dropped immediately. On the opposite, if a packet contains a Group MAC tag whereas the session does not use this feature, this tag MUST be ignored;

- then, verify the digital signature (with TESLA signaling packets) or enter the TESLA authentication process (with data packets).

4.1.1. Processing the Group MAC Tag

Upon receiving a packet containing a Group MAC tag, the receiver recomputes the Group MAC and compares it to the value carried in the packet. If the check fails, the packet MUST be dropped immediately.

More specifically, recomputing the Group MAC requires saving the value of the "Group MAC" field, setting this field to 0, and doing the same computation as a sender does (see Section 3.3.3).

4.1.2. Processing the Digital Signature

Upon receiving a packet containing a digital signature, the receiver verifies the signature as follows.

The computation of the signature MUST include the ALC or NORM header (with the various header extensions) and the payload when applicable. The UDP/IP headers MUST NOT be included. During this computation, the "Signature" field MUST be set to 0 as well as the optional Group MAC, when present.

From [RFC4359]: Digital signature verification is performed as described in [RFC3447], Section 8.2.2 (RSASSA-PKCS1-v1_5) and [RFC3447], Section 8.1.2 (RSASSA-PSS). Upon receipt, the digital signature is passed to the verification function as S. The authenticated portion of the packet is used as the message M, and the RSA public key is passed as (n, e). In summary (when SHA-256 is used), the verification function computes a SHA-256 hash of the authenticated packet bytes, decrypts the SHA-256 hash in the packet, and validates that the appropriate encoding was applied. The two SHA-256 hashes are compared, and if they are identical the validation is successful.

It is assumed that the receivers have the possibility to retrieve the sender's public key required to check this digital signature (Section 2.2). This document does not specify how the public key of the sender is communicated reliably and in a secure way to all possible receivers.

4.1.3. Processing the Authentication Tag

When a receiver wants to authenticate a packet using an authentication tag and when he has the key for the associated time interval (i.e., after the disclosing delay, d), the receiver recomputes the MAC and compares it to the value carried in the packet. If the check fails, the packet MUST be immediately dropped.

More specifically, recomputing the MAC requires saving the value of the "MAC" field, setting this field to 0, and doing the same computation as a sender does (see Section 3.3.1).

4.2. Initialization of a Receiver

A receiver MUST be initialized before being able to authenticate the source of incoming packets. This can be done by an out-of-band mechanism or an in-band mechanism (Section 2.2). Let us focus on the in-band mechanism. Two actions must be performed:

- receive and process a bootstrap information message, and
- calculate an upper bound of the sender's local time. To that purpose, the receiver must perform time synchronization.

4.2.1. Processing the Bootstrap Information Message

A receiver must first receive a packet containing the bootstrap information, digitally signed by the sender. Once the bootstrap information has been authenticated (see Section 4.1), the receiver can initialize its TESLA component. The receiver MUST then ignore the following bootstrap information messages, if any. There is an exception though: when a new key chain is used and if a receiver missed all the commitments for this new key chain, then this receiver MUST process one of the future bootstrap information messages (if any) in order to be able to authenticate the incoming packets associated to this new key chain.

Before TESLA has been initialized, a receiver MUST discard incoming packets other than the bootstrap information message and direct time synchronization response.

4.2.2. Performing Time Synchronization

First of all, the receiver must know whether the ALC or NORM session relies on direct or indirect time synchronization. This information is communicated by an out-of-band mechanism (for instance, when describing the various parameters of an ALC or NORM session). In some cases, both mechanisms might be available and the receiver can choose the preferred technique.

4.2.2.1. Direct Time Synchronization

In the case of a direct time synchronization, a receiver MUST synchronize with the sender. To that purpose, the receiver sends a direct time synchronization request message. This message includes the local time (in NTP timestamp format) at the receiver when sending the message. This timestamp will be copied in the sender's response for the receiver to associate the response to the request.

The direct time synchronization request message format is the following:

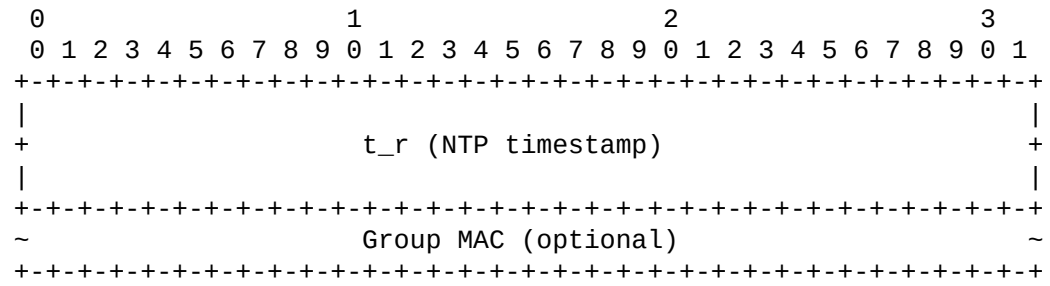

Figure 8: Format of a Direct Time Synchronization Request

The direct time synchronization request (Figure 8) contains the following information:

"t_r" (NTP timestamp, 64 bits):

"t_r" is a timestamp in NTP timestamp format that contains the receiver local time value when sending this direct time synchronization request message;

"Group MAC" field (optional, variable length, multiple of 32 bits):

This field contains the Group MAC, calculated with the group key, K_g, shared by all group members. The field length, in bits, is given by n_w, which is known once the Group MAC function type is known (Section 7).

The receiver then awaits a response message (Section 3.4.2). Upon receiving this message, the receiver:

checks that this response relates to the request, by comparing the "t_r" fields;

checks the Group MAC if present;

checks the signature;

retrieves the t_s value and calculates D_t (Section 2.4.1).

Note that in an ALC session, the direct time synchronization request message is sent to the sender by an out-of-band mechanism that is not specified by the current document.

4.2.2.2. Indirect Time Synchronization

With the indirect time synchronization method, the sender MAY provide out-of-band the URL or IP address of the NTP server(s) he trusts along with an OPTIONAL certificate for each NTP server. When several NTP servers are specified, a receiver MUST choose one of them. This document does not specify how the choice is made, but for the sake of scalability, the clients SHOULD NOT use the same server if several possibilities are offered. The NTP synchronization between the NTP server and the receiver MUST be authenticated, either using the certificate provided by the server or another certificate the client may obtain for this NTP server.

Then the receiver computes the time offset between itself and the NTP server chosen. Note that the receiver does not need to update the local time, (which often requires root privileges), computing the time offset is sufficient.

Since the offset between the server and the time reference, D^O_t, is indicated in the bootstrap information message (or communicated out-of-band), the receiver can now calculate an upper bound of the sender's local time (Section 2.4.2).

Note that this scenario assumes that each client trusts the sender and accepts aligning its NTP configuration to that of the sender, using one of the NTP server(s) suggested. If this assumption does not hold, the client MUST NOT use the NTP indirect time synchronization method (Section 2.3.2).

4.3. Authentication of Received Packets

The receiver can now authenticate incoming packets (other than bootstrap information and direct time synchronization response packets). To that purpose, he MUST follow different steps (see [RFC4082], Section 3.5):

1. The receiver parses the different packet headers. If none of the four TESLA authentication tags are present, the receiver MUST discard the packet. If the session is in "Single Key Chain" mode (e.g., when the "S" flag is set in the bootstrap information message), then the receiver MUST discard any packet containing an Authentication Tag with a New Key Chain Commitment or an Authentication Tag with a Last Key of Old Chain Disclosure.

2. Safe packet test: When the receiver receives packet P_j, it first records the local time T at which the packet arrived. The receiver then computes an upper bound t_j on the sender's clock at the time when the packet arrived: t_j = T + D_t. The receiver

then computes the highest interval the sender could possibly be in: highest_i = floor((t_j - T_0) / T_int). He also retrieves the "i" interval index from the authentication tag. The receiver can now proceed with the "safe packet" test. If highest_i < i + d, then the sender is not yet in the interval during which it discloses the key K_i. The packet is safe (but not necessarily authentic). If the test fails, the packet is unsafe, and the receiver MUST discard the packet.

3. Group MAC test: if the optional Group MAC tag is present and if the session uses this feature, then verify the Group MAC (Section 4.1.1). If the verification fails, the packet MUST be immediately dropped. A packet that does not contain a Group MAC tag whereas the session uses this feature MUST be immediately dropped. On the opposite, if a packet contains a Group MAC tag whereas the session does not use this feature, this tag MUST be ignored.

4. Disclosed Key processing: When the packet discloses a key (i.e., with a Standard Authentication Tag, or with an Authentication Tag with a Last Key of Old Chain Disclosure), the following tests are performed:

   * New key index test: the receiver checks whether a legitimate key already exists with the same index (i.e., i-d). If such a legitimate key exists, the receiver compares its value with the current disclosed key and if they are identical, skips the "Unverifiable key test" and "Key verification test". If such a legitimate key exists but the values differ, the receiver MUST discard the packet.

   * Unverifiable key test: when the disclosed key index is new, it is possible that no earlier disclosed and legitimate key exists for this key chain, thereby preventing the verification of the disclosed key. This happens when the disclosed key belongs to the old key chain and no commitment to this old key chain has ever been received (e.g., because the first bootstrap packet received by a latecomer is for the current key chain, and therefore includes a commitment to the current key chain, not the previous one). When this happens, the receiver MUST ignore the disclosed key (anyway useless) and skip the Key verification test.

   * Key verification test: If the disclosed key index is new and the key can be verified, the receiver checks the legitimacy of K_{i-d} by verifying, for some earlier disclosed and legitimate key K_v (with v < i-d), that K_v and F^{i-d-v}(K_{i-d}) are identical. In other words, the receiver

checks the disclosed key by computing the necessary number of PRF functions to obtain a previously disclosed and legitimate (i.e., verified) key. If the key verification fails, the receiver MUST discard the packet. If the key verification succeeds, this key is said to be legitimate and is stored by the receiver, as well as all the keys between indexes v and i-d.

5. When applicable, the receiver performs any congestion control related action (i.e., the ALC or NORM headers are used by the associated congestion control building block, if any), even if the packet has not yet been authenticated [RFC5651]. If this feature leads to a potential DoS attack (the attacker can send a faked packet with a wrong sequence number to simulate packet losses), it does not compromise the security features offered by TESLA and enables a rapid reaction in front of actual congestion problems.

6. The receiver then buffers the packet for a later authentication, once the corresponding key will be disclosed (after d time intervals) or deduced from another key (if all packets disclosing this key are lost). In some situations, this packet might also be discarded later, if it turns out that the receiver will never be able to deduce the associated key.

7. Authentication test: Let v be the smallest index of the legitimate keys known by the receiver so far. For all the new keys K_w, with v < w <= i-d, that have been either disclosed by this packet (i.e., K_{i-d}) or derived by K_{i-d} (i.e., keys in interval {v+1,.. i-d-1}), the receiver verifies the authenticity of the safe packets buffered for the corresponding interval w. To authenticate one of the buffered packets P_h containing message M_h protected with a MAC that used key index w, the receiver will compute K'_w = F'(K_w) from which it can compute MAC(K'_w, M_h). If this MAC does not equal the MAC stored in the packet, the receiver MUST discard the packet. If the two MACs are equal, the packet is successfully authenticated and the receiver continues processing it.

8. Authenticated new key chain commitment processing: If the authenticated packet contains a new key chain commitment and if no verified commitment already exists, then the receiver stores the commitment to the new key chain. Then, if there are non-authenticated packets for a previous chain (i.e., the key chain before the current one), all these packets can be discarded (Section 4.4).

9. The receiver continues the ALC or NORM processing of all the packets authenticated during the authentication test.

In this specification, a receiver using TESLA MUST immediately drop unsafe packets. But the receiver MAY also decide, at any time, to continue an ALC or NORM session in unsafe (insecure) mode, ignoring TESLA extensions. There SHOULD be an explicit user action to that purpose.

4.3.1. Discarding Unnecessary Packets Earlier

Following strictly the above steps can lead to excessive processing overhead in certain situations. This is the case when a receiver receives packets for an unwanted object with the ALC or NORM protocols, i.e., an object in which the application (or the end user) explicitly mentioned it is not interested. This is also the case when a receiver receives packets for an already decoded object, or when this object has been partitioned in several blocks, for an already decoded block. When such a packet is received, which is easily identified by looking at the receiver's status for the incoming ALC or NORM packet, the receiver MUST also check that the packet is a pure data packet that does not contain any signaling information of importance for the session.

With ALC, a packet containing an "A" flag ("Close Session") or a "B" flag ("Close Object") MUST NOT be discarded before having been authenticated and processed normally. Otherwise, the receiver can safely discard the incoming packet for instance just after step 1 of Section 4.3. This optimization can dramatically reduce the processing overhead by avoiding many useless authentication checks.

4.4. Flushing the Non-Authenticated Packets of a Previous Key Chain

In some cases, a receiver having experienced a very long disconnection might have lost all the disclosures of the last key(s) of a previous key chain. Let j be the index of this key chain for which there remains non-authenticated packets. This receiver can flush all the packets of the key chain j if he determines that:

- he has just switched to a chain of index j+2 (inclusive) or higher;
- the sender has sent a commitment to the new key chain of index j+2 (Section 3.1.2.3). This situation requires that the receiver has received a packet containing such a commitment and that he has been able to check its integrity. In some cases, it might require receiving a bootstrap information message for the current key chain.

If one of the above two tests succeeds, the sender can discard all the awaiting packets since there is no way to authenticate them.

5. Integration in the ALC and NORM Protocols

5.1. Authentication Header Extension Format

The integration of TESLA in ALC or NORM is similar and relies on the header extension mechanism defined in both protocols. More precisely, this document details the EXT_AUTH==1 header extension defined in [RFC5651].

Several fields are added in addition to the "HET" (Header Extension Type) and "HEL" (Header Extension Length) fields (Figure 9).

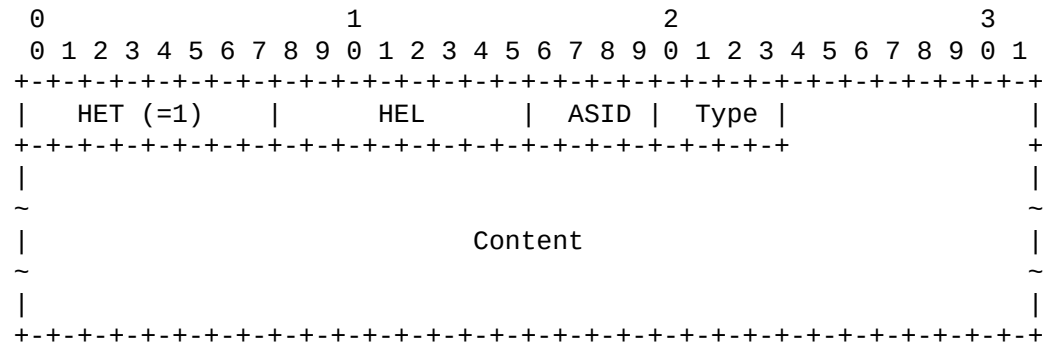

Figure 9: Format of the TESLA EXT_AUTH Header Extension

The fields of the TESLA EXT_AUTH Header Extension are:

"ASID" (Authentication Scheme IDentifier) field (4 bits):

The "ASID" identifies the source authentication scheme or protocol in use. The association between the "ASID" value and the actual authentication scheme is defined out-of-band, at session startup.

"Type" field (4 bits):

The "Type" field identifies the type of TESLA information carried in this header extension. This specification defines the following types:

* 0: Bootstrap information, sent by the sender periodically or after a direct time synchronization request;

* 1: Standard Authentication Tag for the ongoing key chain, sent by the sender along with a packet;

* 2: Authentication Tag without Key Disclosure, sent by the sender along with a packet;

* 3: Authentication Tag with a New Key Chain Commitment, sent by the sender when approaching the end of a key chain;

* 4: Authentication Tag with a Last Key of Old Chain Disclosure, sent by the sender some time after moving to a new key chain;

* 5: Direct time synchronization request, sent by a NORM receiver. This type of message is invalid in the case of an ALC session since ALC is restricted to unidirectional transmissions. Yet, an external mechanism may provide the direct time synchronization functionality;

* 6: Direct time synchronization response, sent by a NORM sender. This type of message is invalid in the case of an ALC session since ALC is restricted to unidirectional transmissions. Yet, an external mechanism may provide the direct time synchronization functionality.

"Content" field (variable length):

This is the TESLA information carried in the header extension, whose type is given by the "Type" field.

5.2. Use of Authentication Header Extensions

Each packet sent by the session's sender MUST contain exactly one TESLA EXT_AUTH Header Extension.

All receivers MUST recognize EXT_AUTH but MAY not be able to parse its content, for instance, because they do not support TESLA. In that case, these receivers MUST ignore the TESLA EXT_AUTH extensions. In the case of NORM, the packets sent by receivers MAY contain a direct synchronization request but MUST NOT contain any of the other five TESLA EXT_AUTH Header Extensions.

5.2.1. EXT_AUTH Header Extension of Type Bootstrap Information

The "bootstrap information" TESLA EXT_AUTH (Type==0) MUST be sent in a stand-alone control packet, rather than in a packet containing application data. The reason for that is the large size of this bootstrap information. By using stand-alone packets, the maximum payload size of data packets is only affected by the (mandatory) authentication information header extension.

With ALC, the "bootstrap information" TESLA EXT_AUTH MUST be sent in a control packet, i.e., containing no encoding symbol.

With NORM, the "bootstrap information" TESLA EXT_AUTH MUST be sent in a NORM_CMD(APPLICATION) message.

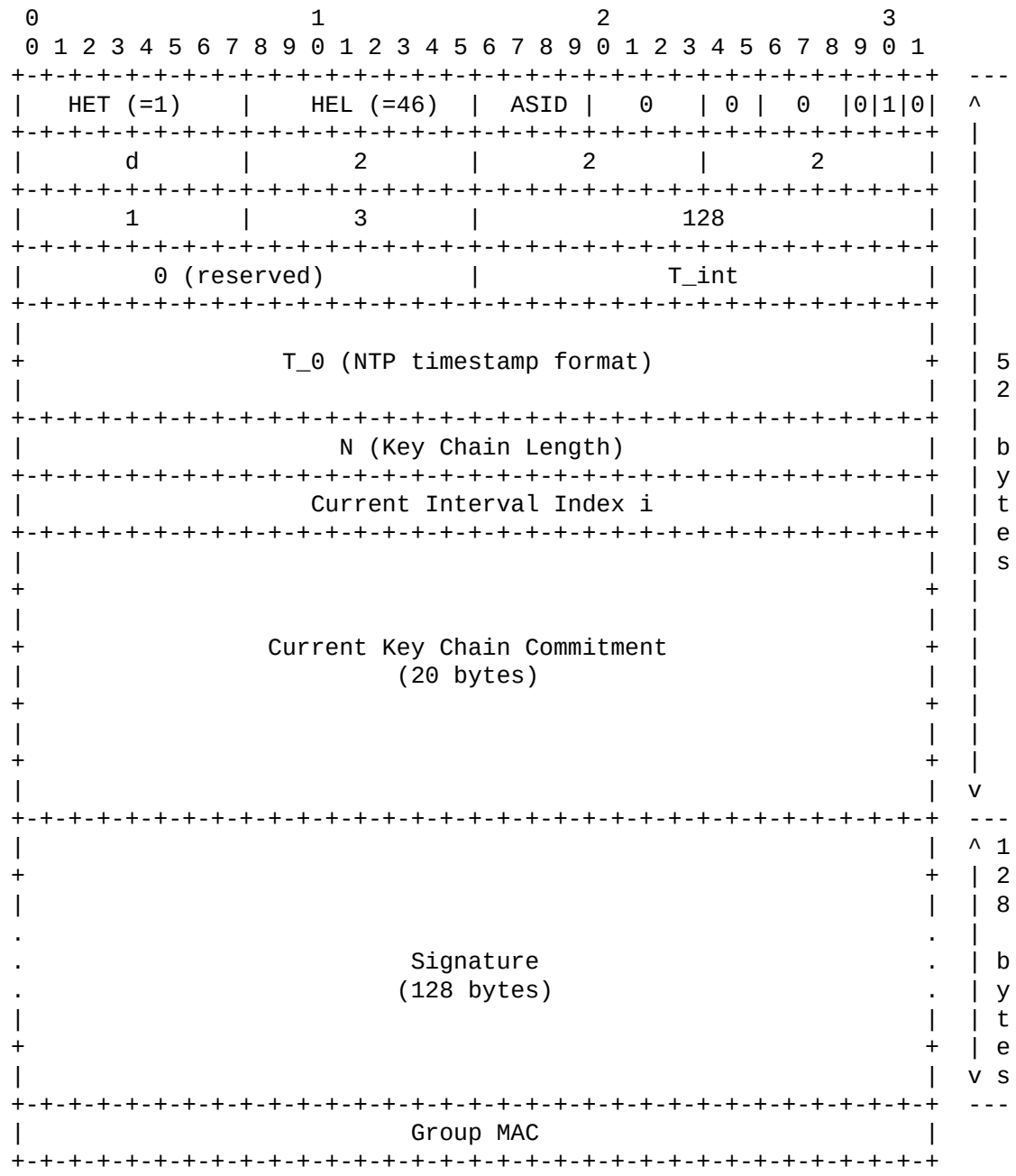

Figure 10: Example: Format of the Bootstrap Information Message (Type 0) Using SHA-256/1024-Bit Signatures, the Default HMAC-SHA-256, and a Group MAC

For instance, Figure 10 shows the bootstrap information message when using the HMAC-SHA-256 transform for the PRF, MAC, and Group MAC functions, along with SHA-256/128 byte (1024 bit) key digital signatures (which also means that the "Signature" field is 128 bytes long). The TESLA EXT_AUTH Header Extension is then 184 bytes long (i.e., 46 words of 32 bits).

5.2.2. EXT_AUTH Header Extension of Type Authentication Tag

The four "authentication tag" TESLA EXT_AUTH Header Extensions (Type 1, 2, 3, and 4) MUST be attached to the ALC or NORM packet (data or control packet) that they protect.

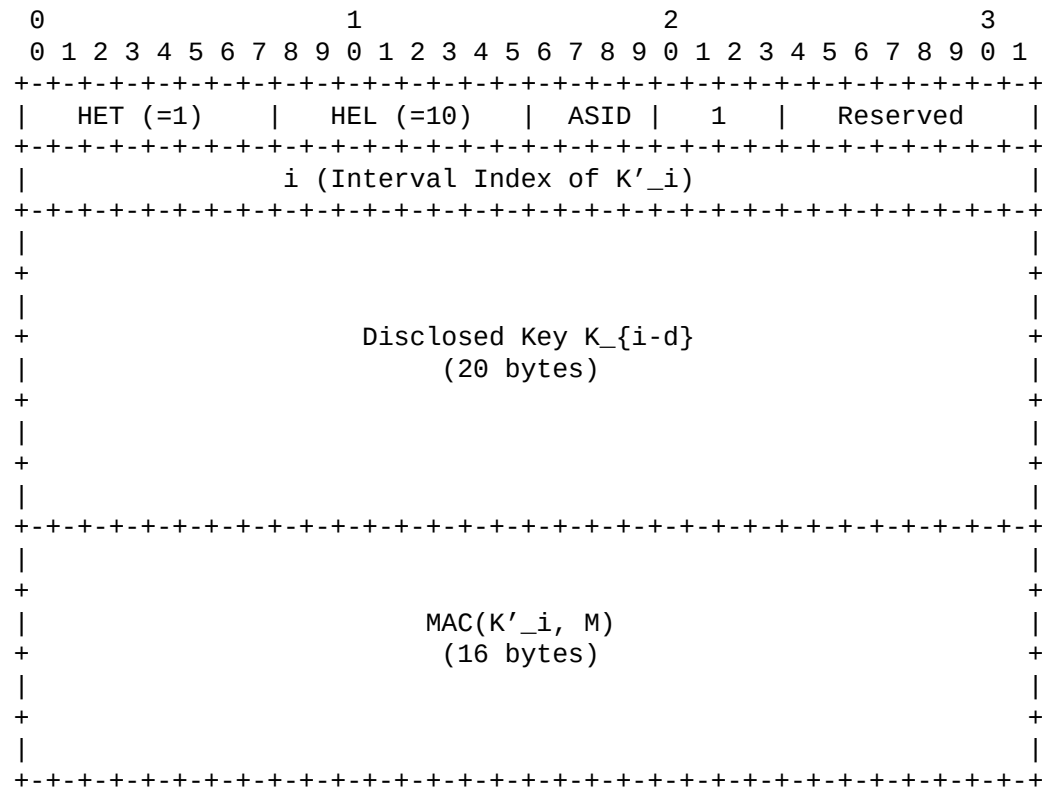

Figure 11: Example: Format of the Standard Authentication Tag (Type 1) Using the Default HMAC-SHA-256

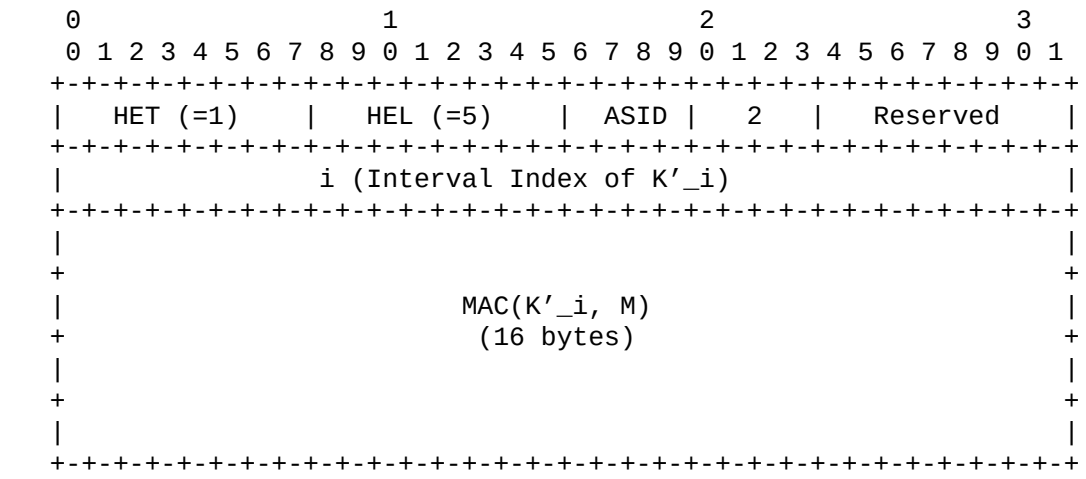

Figure 12: Example: Format of the Authentication Tag without Key Disclosure (Type 2) Using the Default HMAC-SHA-256

For instance, Figures 11 and 12 show the format of the authentication tags, respectively with and without the K_{i-d} key disclosure, when using the (default) HMAC-SHA-256 transform for the PRF and MAC functions. In these examples, the Group MAC feature is not used.

5.2.3. EXT_AUTH Header Extension of Type Direct Time Synchronization Request

With NORM, the "direct time synchronization request" TESLA EXT_AUTH (Type==7) MUST be sent by a receiver in a NORM_CMD(APPLICATION) NORM packet.

With ALC, the "direct time synchronization request" TESLA EXT_AUTH cannot be included in an ALC packet, since ALC is restricted to unidirectional transmissions, from the session's sender to the receivers. An external mechanism must be used with ALC for carrying direct time synchronization requests to the session's sender.

In the case of direct time synchronization, it is RECOMMENDED that the receivers spread the transmission of direct time synchronization requests over the time (Section 2.3.1).

5.2.4. EXT_AUTH Header Extension of Type Direct Time Synchronization Response

With NORM, the "direct time synchronization response" TESLA EXT_AUTH (Type==8) MUST be sent by the sender in a NORM_CMD(APPLICATION) message.

With ALC, the "direct time synchronization response" TESLA EXT_AUTH can be sent in an ALC control packet (i.e., containing no encoding symbol) or through the external mechanism used to carry the direct time synchronization request.

6. Security Considerations

[RFC4082] discusses the security of TESLA in general. These considerations apply to the present specification, namely:

- great care must be taken in the timing aspects. In particular, the D_t parameter is critical and must be correctly initialized;
- if the sender realizes that the key disclosure schedule is not appropriate, then the current session MUST be closed and a new one created. Indeed, Section 3.1.3 requires that these parameters be fixed during the whole session.
- when the verifier that authenticates the incoming packets and the application that uses the data are two different components, there is a risk that an attacker located between these components inject faked data. Similarly, when the verifier and the secure timing system are two different components, there is a risk that an attacker located between these components inject faked timing information. For instance, when the verifier reads the local time by means of a dedicated system call (e.g., gettimeofday()), if an attacker controls the host, he may catch the system call and return a faked time information.

The current specification discusses additional aspects with more details.

6.1. Dealing with DoS Attacks

TESLA introduces new opportunities for an attacker to mount DoS attacks. For instance, an attacker can try to saturate the processing capabilities of the receiver (faked packets are easy to create but checking them requires computing a MAC over the packet or sometimes checking a digital signature as with the bootstrap and direct time synchronization response messages). An attacker can also try to saturate the receiver's memory (since authentication is delayed and non-authenticated packets will accumulate), or to make the receiver believe that a congestion has happened (since congestion control MUST be performed before authenticating incoming packets, Section 4.3).

In order to mitigate these attacks, it is RECOMMENDED to use the Group MAC scheme (Section 3.3.3). No mitigation is possible if a group member acts as an attacker with Group MAC.

Generally, it is RECOMMENDED that the amount of memory used to store incoming packets waiting to be authenticated be limited to a reasonable value.

6.2. Dealing With Replay Attacks

Replay attacks, whereby an attacker stores a valid message and replays it later, can have significant impacts, depending on the message type. Two levels of impacts must be distinguished:

- within the TESLA protocol, and
- within the ALC or NORM protocol.

6.2.1. Impacts of Replay Attacks on TESLA

Replay attacks can impact the TESLA component itself. We review here the potential impacts of such an attack depending on the TESLA message type:

- bootstrap information: Since most parameters contained in a bootstrap information message are static, replay attacks have no consequences. The fact that the "i" and "K_i" fields can be updated in subsequent bootstrap information messages does not create a problem either, since all "i" and "K_i" fields sent remain valid. Finally, a receiver that successfully initialized its TESLA component MUST ignore the following messages (see Section 4.2.1 for an exception to this rule), which voids replay attacks, unless he missed all the commitments to a new key chain (e.g., after a long disconnection) (Section 3.2.1).

- direct time synchronization request: If the Group MAC scheme is used, an attacker that is not a member of the group can replay a packet and oblige the sender to respond, which requires digitally signing the response, a time-consuming process. If the Group MAC scheme is not used, an attacker can easily forge a request anyway. In both cases, the attack will not compromise the TESLA component, but might create a DoS. If this is a concern, it is RECOMMENDED, when the Group MAC scheme is used, that the sender verify the "t_r" NTP timestamp contained in the request and respond only if this value is strictly larger than the previous one received from this receiver. When the Group MAC scheme is not used, this attack can be mitigated by limiting the number of requests per second that will be processed.

- direct time synchronization response: Upon receiving a response, a receiver who has no pending request MUST immediately drop the packet. If this receiver has previously issued a request, he first checks the Group MAC (if applicable), then the "t_r" field, to be sure it is a response to his request, and finally the digital signature. A replayed packet will be dropped during these verifications, without compromising the TESLA component.

- other messages, containing an authentication tag: Replaying a packet containing a TESLA authentication tag will never compromise the TESLA component itself (but perhaps the underlying ALC or NORM component, see below).

To conclude, TESLA itself is robust in front of replay attacks.

6.2.2. Impacts of Replay Attacks on NORM

We review here the potential impacts of a replay attack on the NORM component. Note that we do not consider here the protocols that could be used along with NORM, for instance, the congestion control protocols.

First, let us consider replay attacks within a given NORM session. NORM defines a "sequence" field that can be used to protect against replay attacks [RFC5740] within a given NORM session. This "sequence" field is a 16-bit value that is set by the message originator (sender or receiver) as a monotonically increasing number incremented with each NORM message transmitted. It is RECOMMENDED that a receiver check this "sequence" field and drop messages considered as replayed. Similarly, it is RECOMMENDED that a sender check this sequence, for each known receiver, and drop messages considered as replayed. In both cases, checking this "sequence" field SHOULD be done before TESLA processing of the packet: if the "sequence" field has not been corrupted, the replay attack will immediately be identified; otherwise, the packet will fail the TESLA authentication test. This analysis shows that NORM itself is robust in front of replay attacks within the same session.

Now let us consider replay attacks across several NORM sessions. Since the key chain used in each session MUST differ, a packet replayed in a subsequent session will be identified as unauthentic. Therefore, NORM is robust in front of replay attacks across different sessions.

6.2.3. Impacts of Replay Attacks on ALC

We review here the potential impacts of a replay attack on the ALC component. Note that we do not consider here the protocols that could be used along with ALC, for instance, the layered or wave-based congestion control protocols.

First, let us consider replay attacks within a given ALC session:

- Regular packets containing an authentication tag: a replayed message containing an encoding symbol will be detected once authenticated, thanks to the object/block/symbol identifiers, and will be silently discarded. This kind of replay attack is only penalizing in terms of memory and processing load, but does not compromise the ALC behavior.

- Control packets containing an authentication tag: ALC control packets, by definition, do not include any encoding symbol and therefore do not include any object/block/symbol identifier that would enable a receiver to identify duplicates. However, a sender has a very limited number of reasons to send control packets. More precisely:

  * At the end of the session, a "Close Session" ("A" flag) packet is sent. Replaying this packet has no impact since the receivers already left.

  * Similarly, replaying a packet containing a "Close Object" ("B" flag) has no impact since this object is probably already marked as closed by the receiver.

This analysis shows that ALC itself is robust in front of replay attacks within the same session.

Now let us consider replay attacks across several ALC sessions. Since the key chain used in each session MUST differ, a packet replayed in a subsequent session will be identified as unauthentic. Therefore, ALC is robust in front of replay attacks across different sessions.

6.3. Security of the Back Channel

As specified in Section 1.1, this specification does not consider the packets that may be sent by receivers, for instance, NORM's feedback packets. When a back channel is used, its security is critical to the global security, and an appropriate security mechanism MUST be used. [RMT-SIMPLE-AUTH] describes several techniques that can be used to that purpose. However, the authentication and integrity

verification of the packets sent by receivers on the back channel, if any, is out of the scope of this document.

7. IANA Considerations

IANA has registered the following attributes according to this document. The registries are provided by [RFC4442] under the "Timed Efficient Stream Loss-tolerant Authentication (TESLA) Parameters" registry [TESLA-REG]. Following the policies outlined in [RFC4442], the values in the range up to 240 (including 240) for the following attributes are assigned after expert review by the MSEC working group or its designated successor. The values in the range from 241 to 255 are reserved for private use.

Cryptographic Pseudo-Random Function, TESLA-PRF: All implementations MUST support HMAC-SHA-256 (default).

PRF name	Value
HMAC-SHA1	0
HMAC-SHA-224	1
HMAC-SHA-256 (default)	2
HMAC-SHA-384	3
HMAC-SHA-512	4

Cryptographic Message Authentication Code (MAC) Function, TESLA-MAC: All implementations MUST support HMAC-SHA-256 (default). These MAC schemes are used both for the computing of regular MAC and the Group MAC (if applicable).

MAC name	Value
HMAC-SHA1	0
HMAC-SHA-224	1
HMAC-SHA-256 (default)	2
HMAC-SHA-384	3
HMAC-SHA-512	4

Furthermore, IANA has created two new registries. Here also, the values in the range up to 240 (including 240) for the following attributes are assigned after expert review by the MSEC working group or its designated successor. The values in the range from 241 to 255 are reserved for private use.

Signature Encoding Algorithm, TESLA-SIG-ALGO: All implementations MUST support RSASSA-PKCS1-v1_5 (default).

Signature Algorithm Name	Value
INVALID	0
RSASSA-PKCS1-v1_5 (default)	1
RSASSA-PSS	2

Signature Cryptographic Function, TESLA-SIG-CRYPTO-FUNC: All implementations MUST support SHA-256 (default).

Cryptographic Function Name	Value
INVALID	0
SHA-1	1
SHA-224	2
SHA-256 (default)	3
SHA-384	4
SHA-512	5

8. Acknowledgments

The authors are grateful to Yaron Sheffer, Brian Weis, Ramu Panayappan, Ran Canetti, David L. Mills, Brian Adamson, and Lionel Giraud for their valuable comments while preparing this document. The authors are also grateful to Brian Weis for the digital signature details.

9. References

9.1. Normative References

[RFC1305] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation", RFC 1305, March 1992.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4082] Perrig, A., Song, D., Canetti, R., Tygar, J., and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", RFC 4082, June 2005.

[RFC5651] Luby, M., Watson, M., and L. Vicisano, "Layered Coding Transport (LCT) Building Block", RFC 5651, October 2009.

[RFC5740] Adamson, B., Bormann, C., Handley, M., and J. Macker, "NACK-Oriented Reliable Multicast (NORM) Transport Protocol", RFC 5740, November 2009.

[RFC5775] Luby, M., Watson, M., and L. Vicisano, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 5775, April 2010.

[TESLA-REG] "TESLA Parameters IANA Registry", http://www.iana.org.

9.2. Informative References

[NTP-NTPv4] Burbank, J., Kasch, W., Martin, J., Ed., and D. Mills, "The Network Time Protocol Version 4 Protocol And Algorithm Specification", Work in Progress, October 2009.

[Perrig04] Perrig, A. and J. Tygar, "Secure Broadcast Communication in Wired and Wireless Networks", Kluwer Academic Publishers ISBN 0-7923-7650-1, 2004.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, March 2004.

[RFC4330] Mills, D., "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI", RFC 4330, January 2006.

[RFC4359] Weis, B., "The Use of RSA/SHA-1 Signatures within Encapsulating Security Payload (ESP) and Authentication Header (AH)", RFC 4359, January 2006.

[RFC4383] Baugher, M. and E. Carrara, "The Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Secure Real-time Transport Protocol (SRTP)", RFC 4383, February 2006.

[RFC4442] Fries, S. and H. Tschofenig, "Bootstrapping Timed Efficient Stream Loss-Tolerant Authentication (TESLA)", RFC 4442, March 2006.

[RMT-FLUTE] Paila, T., Walsh, R., Luby, M., Lehtonen, R., and V. Roca, "FLUTE - File Delivery over Unidirectional Transport", Work in Progress, August 2009.

[RMT-SIMPLE-AUTH] Roca, V., "Simple Authentication Schemes for the ALC and NORM Protocols", Work in Progress, October 2009.

Authors' Addresses

Vincent Roca
INRIA
655, av. de l'Europe
Inovallee; Montbonnot
ST ISMIER cedex 38334
France

EMail: vincent.roca@inria.fr
URI: http://planete.inrialpes.fr/~roca/

Aurelien Francillon
INRIA
655, av. de l'Europe
Inovallee; Montbonnot
ST ISMIER cedex 38334
France

EMail: aurelien.francillon@inria.fr
URI: http://planete.inrialpes.fr/~francill/

Sebastien Faurite
INRIA
655, av. de l'Europe
Inovallee; Montbonnot
ST ISMIER cedex 38334
France

EMail: faurite@lcpc.fr