TESLA source authentication in the ALC and NORM protocols

Vincent Roca, Aurelien Francillon, Sebastien Faurite

RMT
Internet-Draft
Expires: January 9, 2006

S. Faurite
A. Francillon
V. Roca
INRIA

July 8, 2005

TESLA source authentication in the ALC and NORM protocols
draft-faurite-rmt-tesla-for-alc-norm-00.txt

Status of this Memo

Copyright Notice

Abstract

This document explains how to integrate the TESLA source authentication and packet integrity protocol to the ALC and NORM content delivery protocols. This draft only considers the authentication/integrity of the packets generated by the session's sender.

Table of Contents

1. Introduction

1.1 Context

This document explains how to integrate the TESLA source authentication and packet integrity protocol to the ALC and NORM protocols. The FLUTE content delivery application, built on top of ALC, can directly benefit from the services offered by TESLA at the transport layer.

This draft only considers the authentication/integrity of the packets generated by the session's sender, not the feedback packets that may be generated by receivers with NORM for instance. Because of the low rate and sporadic transmission of feedback packets, an authentication scheme different from TESLA may be more appropriate in that case. Of course, this remark does not apply to ALC since transmissions are purely unidirectional.

The security offered by TESLA heavily relies on time. Therefore the sender and each receiver need to be time synchronized in a secure way. Two methods exist: a direct one and an indirect one. The present document explains how to achieve time synchronization in each case.

When a broadcaster uses the TESLA with direct time synchronization, each receiver asks the sender for a time synchronization. The source then directly answers to each request, signing the reply. The security of this synchronization method is guaranteed, but the source MAY collapse it the rate of requests exceeds a certain threshold, and a bidirectional channel MUST exist between the source and each receiver. If this may not be an issue with NORM sessions, it will be in case of ALC.

When a broadcaster uses TESLA with indirect time synchronization, the source and each receiver must synchronize with another time reference (e.g. one or more NTP servers, or a GPS device) securely. In that case, if the external time reference does not create a bottleneck, there is no scalability penalty. Besides it works with unidirectional connections from the source to the receiver. Therefore this approach is well suited to ALC sessions.

TESLA requires the transmission of key control information between the source and each receiver. In particular it defines:

- a bootstrapping information message that carries control information needed by a receiver to initialize its TESLA component. Depending on the time synchronization method, this message will be sent either directly to a receiver upon request or

broadcast periodically. This latter possibility enables in particular late receivers to catch up, which is required with ALC sessions in ``on-demand'' mode;

- an authentication tag is associated to each packet in order to enable the authentication of previously sent packets, and possibly to announce a new or old key commitment;

- a direct synchronization request message;

The present document explains how to carry this information in the ALC and NORM protocols, using a dedicated authentication protocol header extension, EXT_AUTH=1 (already mentioned in [RFC3451]).

For more informations on the TESLA protocol and its principles, please refer to [RFC4082] and the references mentioned in that document.

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 are used throughout this document:

- PRF is the Pseudo Random Function.

- MAC is the Message Authentication Code.

- HMAC is the Keyed-Hash Message Authentication Code.

- TN is the reference time at some point.

- T_s is the TESLA server local time corresponding to reference time TN.

- T_r is the TESLA receiver local time corresponding to reference time TN.

- N_tx_old_kcc is ...

- N_tx_new_kcc is ...

- N is the number of keys of a key chain. When several chains are used, all chains have the same length.

- o
- o TODO: finish this list...

2. Time Synchronization

The security of TESLA relies on time. The sender uses a set of keys to compute the MAC of the messages and discloses these keys later. The receiver MUST ensure that each packet he received cannot have been sent after the disclosure of the corresponding key.

To achieve this, the sender and the receiver MUST be synchronized. More precisely, the receiver must know an upper bound of the sender's local time. There are two possibilities:

- Direct time synchronization: each receiver asks the sender for a time synchronization.
- Indirect time synchronization: each receiver and the sender use an other time reference, like a NTP server or GPS device.

2.1 Direct Time Synchronization

The receiver sends a synchronization request, that includes a nonce, to the sender and remembers the local receiver time the message was sent, t_r. The sender records its local time upon receiving the request, t_s. The sender sends a reply (a bootstrapping information message, Section 3.2.1) that includes in particular t_s. This answer, plus the nonce that is appended, is digitally signed. The nonce is then removed, and the reply is sent to the receiver. After authenticating the reply, the receiver can estimate an upper bound of the sender's time: D_t = t_s - t_r + S, where S is an estimated bound on the clock drift throughout the duration of the session. See [RFC4082] for further details.

Since a bidirectional channel is assumed, this method is well suited to NORM sessions. Section 4.1.2 details the request message in case of NORM. Section 3.2.1 details the bootstrapping information.

2.2 Indirect Time Synchronization

Indirect time synchronization relies on a time reference, for instance a NTP time server or a GPS synchronized clock. The sender and each receiver then synchronize directly and independently with this third party. In the case of a synchronization through a NTP time server, several time servers SHOULD be available for scalability purpose.

There are two drawbacks:

- The session's sender looses the control of the synchronization. Since it is vital to the whole security, this synchronization must be fully secured.

- When different NTP servers are used, these servers MUST be securely and reliably synchronized between each other.

2.2.1 Delay Bound Calculation in Indirect time Synchronization

Let's assume the sender and each receiver synchronize with one or several reference time servers. A direct time synchronization between the server and the reference time server results in ds = TN - Ts. A direct time synchronization between a receiver and the same reference time server results in dr = TN - Tr. So: D = ds - dr, is the offset between the TESLA server and the TESLA receiver (since D = (TN - Ts) - (TN - Tr) = Tr - Ts). When a receiver receives packets, he is now able to compute the upper bound on the time server when it sends it. If tr is the time when the receiver gets the packet and ts is the corresponding time of the server, we have tr = Tr + t and ts = Ts + t. The receiver can compute the server time by doing tr - D = (Tr + t) - (Tr - Ts) = ts.

For more security, we can substract to D two positive values:

- an upper bound on the maximum clock drift between the sender and the receiver during the session.

- when the server and the receiver use different NTP time servers : an upper bound on the NTP time error between two different NTP servers.

Let Derr be the sum of these upper bounds.

So: D = ds - dr - Derr, with Derr > 0. When computing tr - D = ts + Derr, we now have a security margin when we do the TESLA security check, that is we must have received the packet before the disclosing of the key used to compute the MAC of this packet.

In the bootstrapping information (see Section 3.3.1), the ds value (which can be negative) is sent. This information does not depend on the receiver, so the bootstraping information can be broadcast to all receivers.

3. Sender Operations

3.1 TESLA Parameters

3.1.1 Time Interval Schedule

The sender must choose a time interval schedule. That is it must decide when the different keys needed by TESLA will be used and disclosed. The different parameters are:

- T_int: the interval duration usually ranging from 100 milliseconds to 1 second.
- d: the key disclosure delay. It is the time to wait (in a number of intervals) to disclose a key. A key is part of a one-way key chain.
- N: the length of the key chain. When a key chain is finished it is possible to switch to a new key chain.
- TI_0: the starting time of time interval 0.

3.1.2 Key Chain

The TESLA sender must compute a one-way key chain of N keys. It must first select a Primary Key, choose two PRF function F and F', and then compute all the previous keys using K_{i-1} = F(K_i). The key for MAC calculation can then be derived from the corresponding K_i key by K'_i=F'(K_i). The randomness of the Primary key is vital to the security since no one should be able to guess it.

The key chain has a finite length, so the TESLA session must finish before the end of the key chain. 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 when necessary.

To do so, the sender must send a commitment to the new key chain before the end of the current key chain. This commitment is simply F(K_{N+1}) and should be sent during N_tx_new_kcc intervals before the end of the key chain. Generally, several packets are sent during a time interval, so it is possible to alternate between sending a disclosed key and a commitment to the new key chain, which offers the benefit of adding no overhead. See Section 3.3.3 for more informations.

The receiver will keep the commitment, until the key K_{N+1} (the first of the new key chain) is disclosed. 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 a 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. This is done during N_tx_old_kcc intervals at the beginning of the new key chain. See Section 3.3.4 for more informations.

3.2 TESLA Signaling

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

- The bootstrap information, which is a digitally signed packet containing all the information required to bootstrap TESLA at a receiver.

- The authentication tag, which is sent in all packets (see Section 5 for exceptions) and contains the MAC of the packet.

3.2.1 Bootstrap Information

In order to initialize the TESLA component at a receiver, the sender must communicate some key information. This TESLA bootstrap information MUST be securely transmitted, in particular a receiver must be able to check the packet source and the packet integrity using standard protocols. Any digital signature will do.

The bootstrap information can be either sent in point to point, after a direct synchronization request from a receiver, or broadcast to all receivers, for instance periodically, when indirect time synchronization is used.

The periodic transmission of the bootstrap information message will be required in indirect time synchronization mode when:

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

- the packet containing the bootstrap information has been lost by some clients,

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 frequency of 1 second for the

transmission of this bootstrap information is often a reasonable value.

The bootstrap information message MAY be sent only once in other cases, in particular with sessions in ``push'' mode, when all clients will receive with a very high probability the corresponding packet.

In some cases, a change in the key chain can lead a receiver having experienced a very long disconnection to loose the commitment of the new chain, and therefore be unable to authenticate any packet related to the new chain and all the following ones. To solve this issue, the sender can extend the T_tx_new_kcc value, but this will remain a partial solution (a receiver may be disconnected during the whole key chain period), or decide to periodically send the bootstrap information message, even in direct time synchronization mode.

3.2.2 Authentication Tag

Every authenticated packet must have an authentication tag, containing the MAC of the message and a disclosed key.

The computation of the MAC, MAC(K_i, M), includes the ALC or NORM header, including the various header extensions, plus the payload when applicable. The UDP/IP/MAC headers are not included. During this computation, the MAC(K_i, M) field of the authentication tag (see Section 3.3.2 Section 3.3.3 Section 3.3.4) MUST be set to 0.

3.3 Signaling Information Format

This section specifies the format of the various kinds of TESLA signaling information sent by the sender.

3.3.1 Bootstrap Information Format

----- Editor's note: This bootstrap information format is based on the expired draft [tesla/spec], modified. The present format is not fully satisfying and will probably be changed in new versions of this I-D. -----

The format of the bootstrap information:

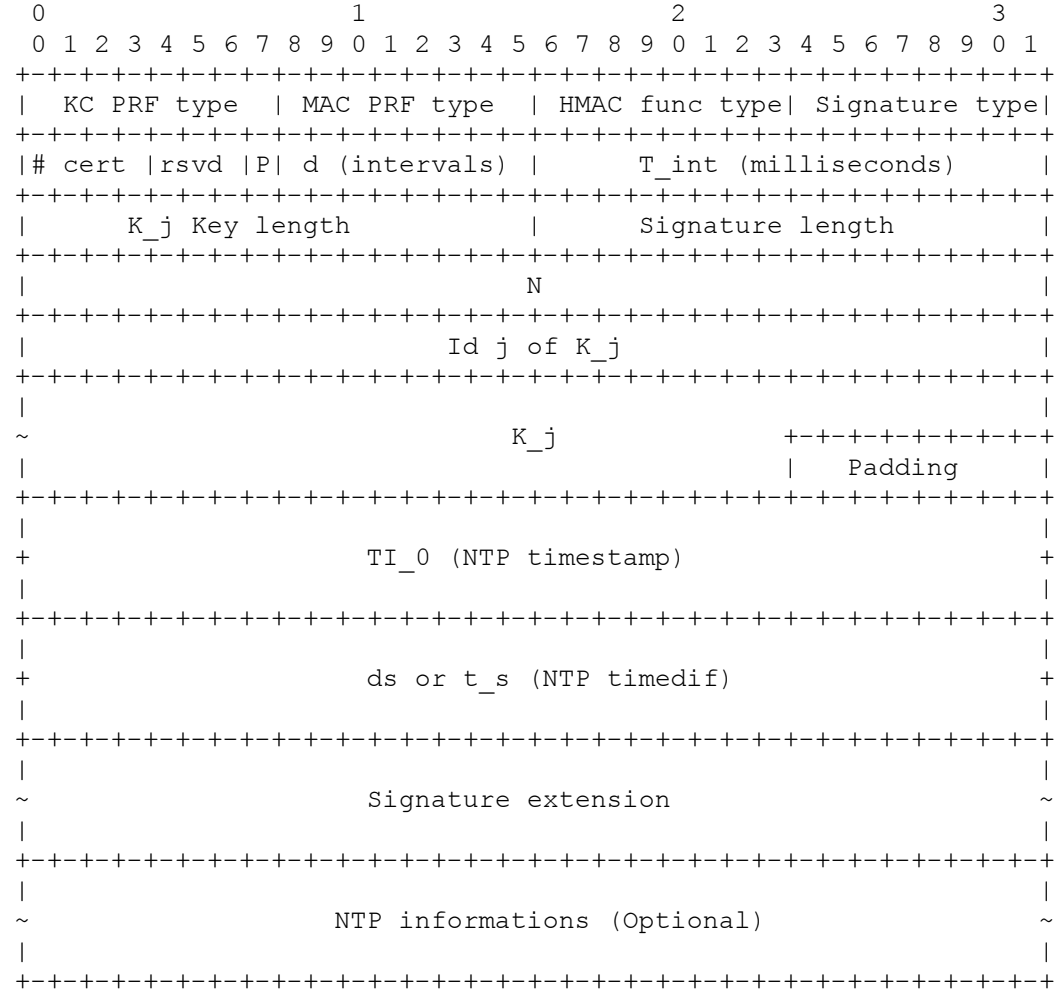

Figure 1

The reference numbers are described in Appendix A

- The key chain PRF type is the reference number of the F function used to calculate the key chain.

- The MAC PRF type is the reference number of the F' function used to derive the MAC key from the key chain.

- The HMAC function type is the reference number of the function used to compute the HMAC of the packets.

- Signature type is the reference number of the digital signature used to authenticate this bootstrap information.
- # of certs is the number of certificates present in the signature extension.
- P ("Positive") is a boolean used in the indirect time synchronization. It indicates whether the D time difference is positive (P=1) or negative (P=0).
- d is an unsigned integer that defines the number of intervals before key disclosure (e.g. if a key is used in interval i, this key will be disclosed in interval i+d). d MUST be greater or equal to 2.
- T_int is an unsigned integer corresponding to the number of milliseconds of one interval.
- K_j Key length is the length in bits of key K_j.
- Signature length is the number of bytes of the signature included in the signature extension.
- N is the number of keys of the key chain.
- Id j of K_j is an unsigned integer corresponding to the index of the interval of the key released in this bootstrap information. For performance reasons, the sender SHOULD always send a bootstrap information with the highest Id j possible since this will reduce the number of computation for the receivers that join later.
- K_j is the key corresponding to the interval j. If i is the current interval we MUST have: j < i - d.
- TI_0 is the time of the beginning of interval 0. It is a NTP timestamp, composed of two 32 bits word: one for the seconds elapsed since 01/01/1900 at 00:00 and the other one for the fraction of seconds.
- ds or t_s (NTP timedif) depends on the time synchronization mode. In direct mode, this field contains t_s, which is the sender's local time when he received the request (Section 2.1). In indirect mode, this field contains ds, the offset between the server and an NTP server or an other time reference (Section 2.2).
- The format of the signature extension is described below, and depends on the "# of certs" field.

- The NTP information is optional and is described below. Its presence can be detected by the total length of the signature.

The signature extension format when the "# of certs" field is strictly greater than 0 (2 in this example) is:

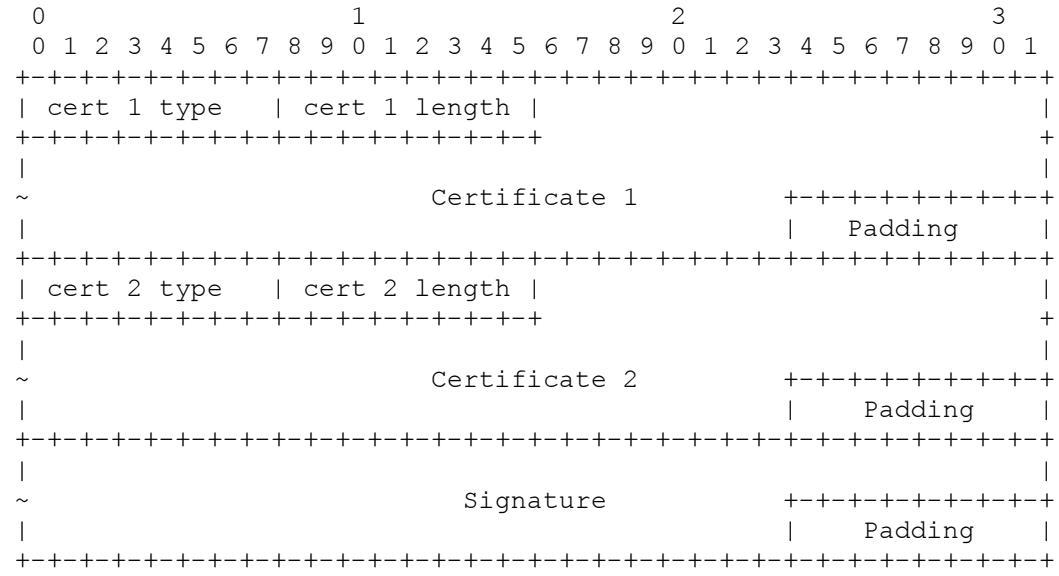

Figure 2

In Figure 2:

- Type of certificate identifies the algorithm used for the certificate (see Appendix A).

- The certificate length is the length in bytes of the certificate.

- The certificate field contains a certificate signed by an external authority and that certifies the sender's public key. This field is padded (with 0) up to a multiple of 32 bits.

- The signature is a digital signature using the type and length specified in the main part of the bootstrap information message. This field is padded (with 0) up to a multiple of 32 bits.

The signature extension format when the "# of certs" field is zero (i.e. when no certificate is provided) is:

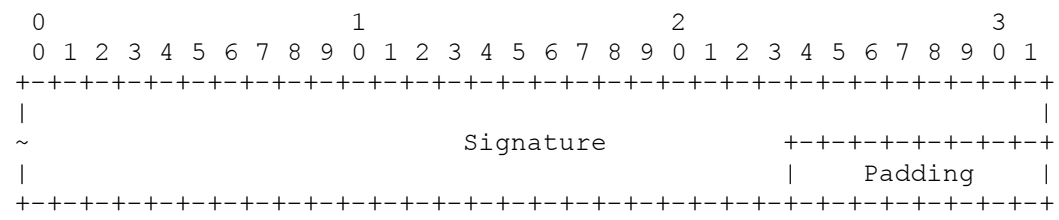

Figure 3

In Figure 3:

- The signature is a digital signature using the type and length specified in the main part of the bootstrap information message. This field is padded (with 0) up to a multiple of 32 bits.

The optional NTP information format, when two NTP servers are specified, is:

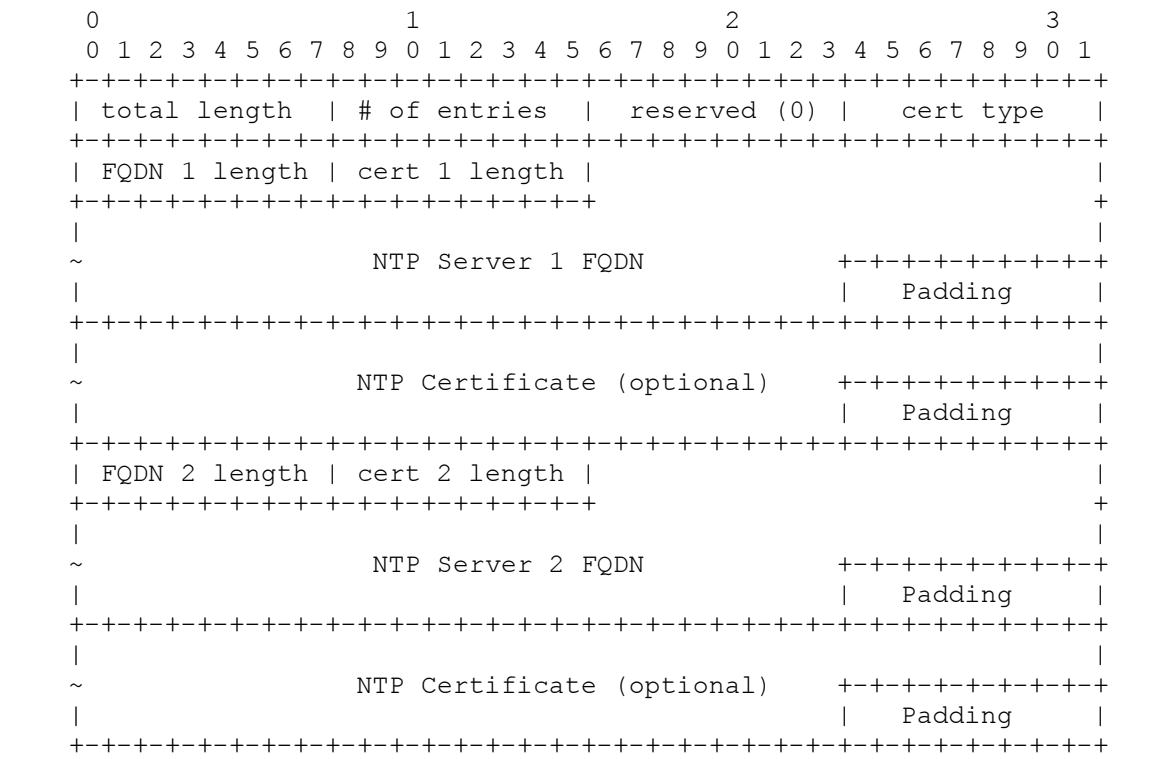

Figure 4

In Figure 4:

- The total length is the total length in units of 32 bit words of this NTP information extension;
- The # of entries is the number of NTP entries;
- Type of certificates identifies the algorithm used for all the certificates that may be provided (see Appendix A).
- The FQDN length is the number of bytes of the NTP server fully qualified domain name;
- The NTP server FQDN is a string containing the NTP server Fully Qualified Domain Name (e.g. "ntp.foo.bar."). This field is padded (with 0) up to a multiple of 32 bits;

o The NTP Certificate is optional. The content delivery server can use it to self-certify the NTP public key. The certificate length indicates whether this field is present or not. This field is padded (with 0) up to a multiple of 32 bits.

3.3.2 Standard Authentication Tag Format

Here is the format of the authentication tag:

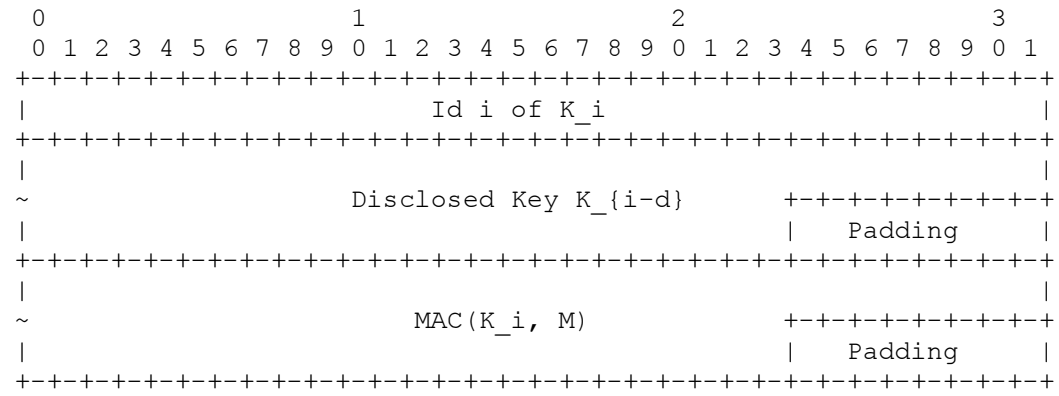

Figure 5

In Figure 5:

o The Id i is the index of the key used for computing the MAC of this packet.

o The disclosed key MUST be the key used for interval i-d.

o MAC(K_i, M) is the message authentication code of the current packet, including the ALC or NORM header (including the header extensions), plus the payload when applicable.

3.3.3 Authentication Tag Format with a New key Chain Commitment

During the last N_tx_new_kcc intervals of the current key chain, the sender MUST send a commitment to the next key chain. This is done by replacing the disclosed key of the authentication tag with the new key commitment, F(K_{N+1})

Here is the format of the authentication tag with an new key commitment tag:

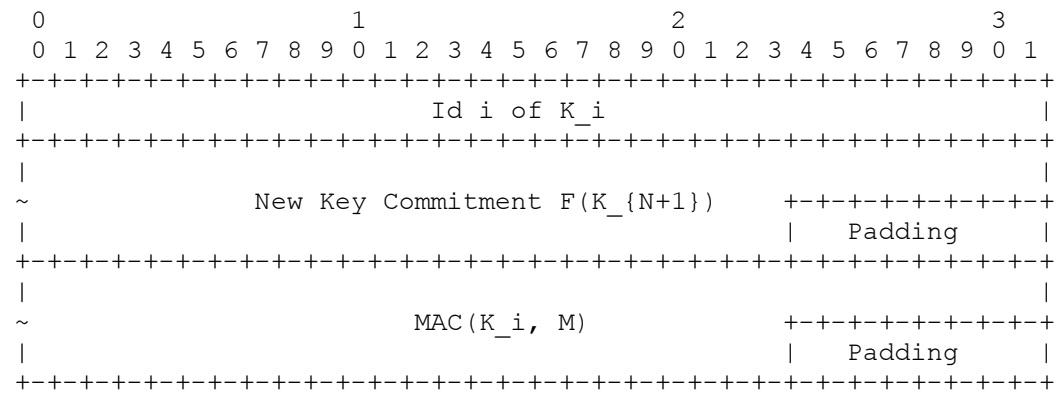

Figure 6

3.3.4 Authentication Tag Format with an Old Key Chain Commitment

During the first N_tx_old_kcc intervals of the new key chain after the disclosing interval, d, the sender must send a commitment to the old key chain. This is done by replacing the disclosed key of the authentication tag with the old key commitment, K_N

Here is the format of the authentication tag with an old key commitment tag:

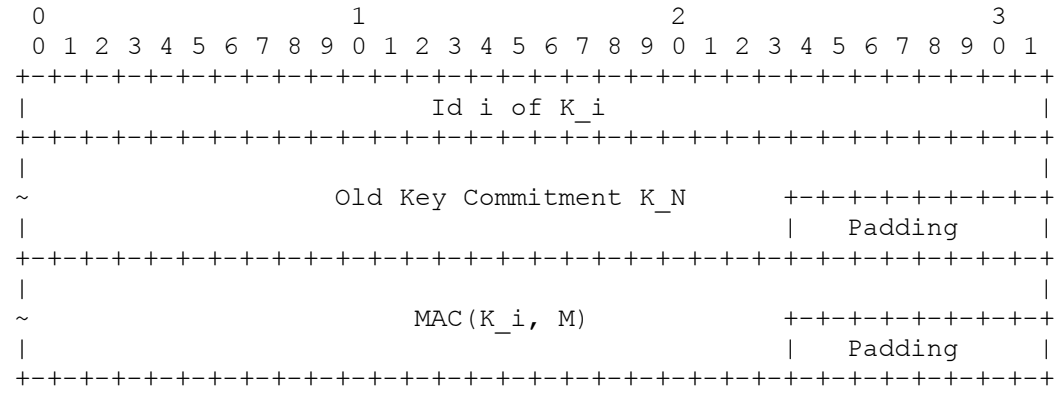

Figure 7

4. Receiver Operations

4.1 Initialization of a Receiver

A receiver must be initialized before being able to authenticate the source of incoming packets. Two actions must be performed:

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

4.1.1 Processing the Bootstrap Information Message

A receiver must receive a packet containing the bootstrap information, digitally signed by the sender, and verify its signature. Because the packet is signed, the receiver also needs to know the public key of the sender. The present document does not specify how the public key of the sender is communicated reliably and in a secure way to all possible receivers. Once the bootstrap information has been proved to be safe, the receiver can initialize its TESLA component. Time synchronization is detailed in Section 4.1.2. The receiver stores the parameters related to the time interval schedule and key chain. The receiver can then ignore next bootstrap information messages, except if he is in a new key chain and missed all the commitments for this new key chain.

Before TESLA has been initialized, a receiver MUST ignore all packets other than the bootstrap information message. Yet, a receiver MAY buffer incoming packets, recording the reception time of each packet, and proceed with delayed authentication later, once the receiver will be fully initialized. In that case, the buffer must be carefully sized.

4.1.2 Time Synchronization

First of all, the receiver must know whether the ALC or NORM session relies on direct or indirect synchronization. This information is communicated by an out-of-band mechanism (for instance when describing the various parameters of a FLUTE session in case of ALC).

In case of a direct time synchronization, a receiver MUST first synchronize with the sender. To that purpose, the receiver sends direct time synchronization request message. This message includes a nonce, i.e. a random number chosen independently by the receiver, that will be integrated when the sender calculates the digital signature of his reply.

The request for a direct time synchronization contains the following information:

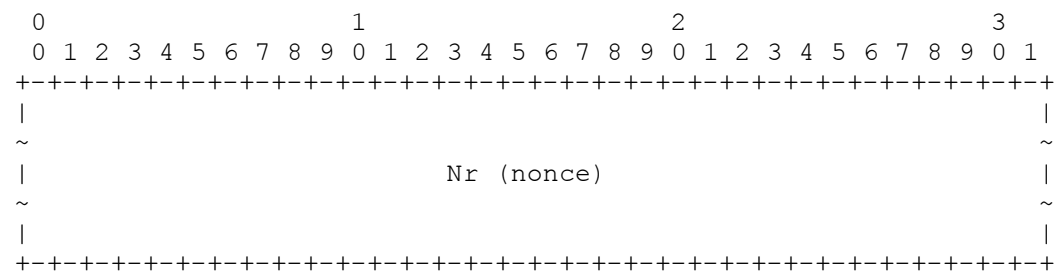

Figure 8

With the indirect time synchronization method, the sender MAY provide in its bootstrap information, the URL of the NTP servers he trusts along with an OPTIONAL certificate for each NTP server. When NTP servers are specified, a receiver SHOULD choose one of the NTP servers provided. 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 secured, either using the certificate provided by the content delivery 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, since this operation would often require some privileges, computing the time offset is sufficient.

Since the offset between the server and the time reference is indicated in the bootstrap information message, the receiver can now calculate an upper bound of the sender's local time Section 2.2.

4.2 Authentication of Received Packets

The receiver can now authenticate incoming packets. To that purpose, he must follows different steps:

1. The receiver parses the different packet headers. If the TESLA authentication tag is not present, the receiver MUST reject the packet.

2. Then proceed with the TESLA safe test: (1) check that the key used to compute the MAC of this packet has not already been disclosed, and (2) check the disclosed key by computing the

necessary number of PRF functions to obtain a previously safe disclosed key. If any of these two tests fail, the receiver MUST reject the packet.

3. Then, according to the [RFC3451], when applicable, perform congestion control even if the packet has not yet been authenticated. If this feature leads to a potential DoS attack, it does not compromise the security features offered by TESLA and enables a rapid reaction in front of congestion problems.

4. Then buffer the packet for a later authentication, once the corresponding key will be received or deduced from another key.

5. If the disclosed key is a new one, then the receiver can authenticate previously stored packets using this key or any key derived from this one.

6. If a packet fails to be authenticated, then this packet MUST be rejected.

7. If a packet is successfully authenticated, then the receiver continues processing it.

----- Editor's note: [RFC4082] explains that unauthenticated packets SHOULD be destroyed, and if not this is at the own risk of the receiver. We choose the other strategy, requiring that unsafe packets be destroyed when the client decides to use TESLA. But the client can at any time choose to continue an ALC or NORM session in unsafe mode, ignoring TESLA extensions. -----

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 we further specify the EXT_AUTH=1 header extension [RFC3451]. Several fields are added in addition to the HET (Header Extension Type) and HEL (Header Extension Length) fields (Figure 9).

Here is the format of the LCT or NORM extension header:

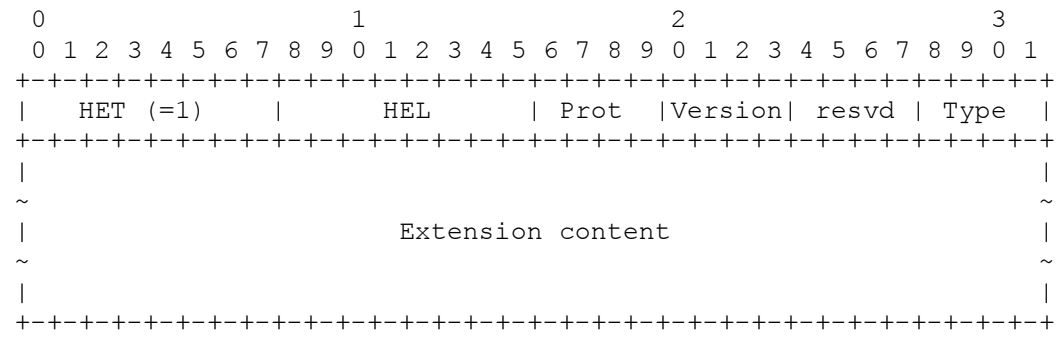

Figure 9

The following fields are defined:

- the Prot (Authentication Protocol) field (4 bits) identifies the source authentication protocol in use. We use 1 for TESLA.
- the Version field (4 bits) identifies the version number of the authentication scheme.
- the resvd (Reserved) field (4 bits) is not used and must be zero'ed.
- the Type field(4 bits) identifies the type of message:
  - 1: bootstrap information, sent by the sender periodically (Section 3.3.2);
  - 2: authentication information for the on-going key chain, sent by the sender along with each packet;
  - 3: authentication information along with a new key chain commitment, sent by the sender when approaching the end of a key chain;

* 4: authentication information along with an old key chain commitment, sent by the sender some time after moving to a new key chain;

* 5: direct synchronization request, sent by a NORM receiver;

Each packet sent by the sender MUST contain exactly one of these header extensions when TESLA is used in an ALC or NORM session. All receivers MUST recognize EXT_AUTH but MAY NOT be able to parse its content, for instance because they do not use the TESLA building block, and in that case they SHOULD ignore the EXT_AUTH extensions. In case of NORM, the packets sent by receivers MAY contain a direct synchronization request but MUST NOT contain any of the other four authentication header extensions.

----- Editor's note: this document defines a "Scheme" field that further identifies the authentication protocol. By doing so, all the documents specifying another authentication protocol and relying on the header extension mechanism MUST reserve the same 4 bit "Scheme" field. One advantage is that several authentication protocols can be used in a session (perhaps with a NORM session for the feedback messages). This possibility must be discussed. The other solution is to say that there's a single authentication protocol per session, communicated out of band, and therefore the authentication header extension is fully defined (same approach as that of the CCI ALC/LCT congestion control field). -----

5.2 Use of Authentication Header Extensions

The bootstrap information (Type=1) SHOULD be sent in a stand-alone control packet rather than in data packets. The reason is the large size of this bootstrap information which largely increases the maximum ALC/LCT or NORM header size. By having the bootstrap information header extension in stand-alone packets, the maximum payload of data packets is only affected by the unavoidable authentication tag, not by an additional large header extension sent at a low frequency.

The three authentication information extension headers (Type=2, 3, or 4) are attached to the corresponding packet (data or control packet). There is no authentication information header extension in case of a control packet containing only the bootstrap information.

In case of NORM, the direct synchronization request extension header (Type=5) is sent by a receiver in a XXX NORM packet (see editor's note below). There is no authentication information header extension in this case since this draft only considers the authentication/

integrity of the packets generated by the session's sender.

----- Editor's note: what type of NORM packet should be used to that purpose? NORM_REPORT is one possibility. TBD... -----

6. Security Considerations

The security of the TESLA protocol is discussed in [RFC4082]. Security considerations specific to its use in ALC and NORM remain TBD...

7. References

7.1 Normative References

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

[RFC3450] Luby, M., Gemmell, J., Vicisano, L., Rizzo, L., and J. Crowcroft, "Asynchronous Layered Coding (ALC) Protocol Instantiation", RFC 3450, December 2002.

[RFC3451] Luby, M., Gemmell, J., Vicisano, L., Rizzo, L., Handley, M., and J. Crowcroft, "Layered Coding Transport (LCT) Building Block", RFC 3451, December 2002.

[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.

7.2 Informative References

[tesla/spec] Perrig, A., Canetti, R., and B. Whillock, "TESLA: Multicast Source Authentication Transform Specification (draft-ietf-msec-tesla-spec-00.txt)", October 2002.

Authors' Addresses

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

Phone:
Email: sebastien.faurite@inrialpes.fr
URI:

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

Phone:
Email: aurelien.francillon@inrialpes.fr
URI:

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

Phone:
Email: vincent.roca@inrialpes.fr
URI:

Appendix A. IANA Considerations

This document requires an IANA registration for the following attributes:

A.1 Cryptographic pseudo-random function (PRF)

We use :

	algorithm
1	MD5

A.2 Cryptographic message authentication code (MAC)

We use :

	algorithm	key length
1	MD5	64
2	MD5	96
3	MD5	128

A.3 Signature type

We use :

	algorithm
1	PKCS #1: RSA Cryptography Standard

A.4 Certificate type

We use :

	algorithm
1	PKCS #1: RSA Cryptography Standard

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.