Internet Engineering Task Force IETF MSEC
Internet Draft
Perrig, Canetti, Song, Tygar, Briscoe
UC Berkeley / Digital Fountain / IBM / BT
draft-ietf-msec-tesla-intro-02.txt
May 2004
Expires: November 2004

TESLA: Multicast Source Authentication Transform Introduction

STATUS OF THIS MEMO

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet Drafts.

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

To view the list Internet-Draft Shadow Directories, see http://www.ietf.org/shadow.html.

Abstract

Data authentication is an important component for many applications, for example audio and video Internet broadcasts, or data distribution by satellite. This document introduces TESLA, a secure source authentication mechanism for multicast or broadcast data streams. This document provides an algorithmic description of the scheme for informational purposes, and in particular, it is intended to assist in writing standardizable and secure specifications for protocols based on TESLA in different contexts.

The main deterrents so far for a data authentication mechanism for multicast were the seemingly conflicting requirements: loss tolerance, high efficiency, no per-receiver state at the sender. The problem is particularly hard in settings with high packet loss rates and where lost packets are not retransmitted, and where the receiver wants to authenticate each packet it receives.

TESLA provides authentication of individual data packets, regardless of the packet loss rate. In addition, TESLA features low overhead for

both sender and receiver, and does not require per-receiver state at the sender. TESLA is secure as long as the sender and receiver are loosely time synchronized.

Table of Contents

1 Introduction

The power of multicast is that one packet can reach millions of receivers. This great property is unfortunately also a great danger: an attacker that sends one malicious packet can also potentially reach millions of receivers. Receivers need multicast source authentication to ensure that a given packet originates from the correct source.

In unicast communication, we can achieve data authentication through a purely symmetric mechanism: the sender and the receiver share a secret key to compute a message authentication code (MAC) of all communicated data. When a message with a correct MAC arrives, the

receiver is assured that the sender generated that message. Standard mechanisms achieve unicast authentication this way, for example TLS or IPsec [1,2].

The symmetric MAC authentication is not secure in a broadcast setting. Consider a sender that broadcasts authentic data to mutually untrusting receivers. The symmetric MAC is not secure: every receiver

knows the MAC key, and hence could impersonate the sender and forge messages to other receivers. Intuitively, we need an asymmetric mechanism to achieve authenticated broadcast, such that every receiver can verify the authenticity of messages it receives, without being able to generate authentic messages. Achieving this in an efficient way is a challenging problem [3].

The standard approach to achieve such asymmetry for authentication is to use asymmetric cryptography, for instance a digital signature. Digital signatures have the required asymmetric property: the sender generates the signature with its private key, and all receivers can verify the signature with the sender's public key, but a receiver with the public key alone cannot generate a digital signature for a new message. A digital signature provides non-repudiation, which is a stronger property than authentication. Unfortunately, digital signatures have a high cost: they have a high computation overhead for both the sender and the receiver, as well as a high communication overhead. Since we assume broadcast settings where the sender does not retransmit lost packets, and the receiver still wants to immediately authenticate each packet it receives, we would need to attach a digital signature to each message. Because of the high overhead of asymmetric cryptography, this approach would restrict us to low-rate streams, and to senders and receivers with powerful workstations. To deal with the high overhead of asymmetric cryptography, we can try to amortize one digital signature over multiple messages. However, such an approach is still expensive in contrast to symmetric cryptography, since symmetric cryptography is in general 3 to 5 orders of magnitude more efficient than asymmetric cryptography. In addition, the straight-forward amortization of one digital signature over multiple packets requires reliability, as the receiver needs to receive all packets to verify the signature. A number of schemes that follow this approach are [4,5,6,7,8]. See [9] for more details.

This document presents the Timed Efficient Stream Loss-tolerant Authentication protocol (TESLA). TESLA uses mainly symmetric

cryptography, and uses time delayed key disclosure to achieve the required asymmetry property. However, TESLA requires loosely synchronized clocks between the sender and the receivers. See more details in Section 4. Other schemes that follow a similar approach to TESLA are [10,11,12].

1.1 Notation

To denote the subscript or an index of a variable, we use the underscore between the variable name and the index, e.g. the key K with index i is K_i, the key K with index i+d is K_{i+d}. To write a superscript we use the caret, e.g. the function F with the argument x executed i times is F^i(x), executed j-1 times we write F^{j-1}(x).

2 Functionality

TESLA provides delayed per-packet data authentication. The key idea to providing both efficiency and security is a delayed disclosure of keys. The delayed key disclosure results in an authentication delay. In practice, the delay is on the order of one RTT (Round-trip-time).

TESLA has the following properties:

- Low computation overhead for generation and verification of authentication information

- Low communication overhead

- Limited buffering required for the sender and the receiver, hence timely authentication for each individual packet

- Strong robustness to packet loss

- Scales to a large number of receivers

- Security is guaranteed as long as the sender and recipients are loosely time synchronized, where synchronization can take place at session set-up.

TESLA can be used either in the network layer, or in the transport layer, or in the application layer. The delayed authentication,

however, requires buffering of packets until authentication is completed.

2.1 Threat Model and Security Guarantee

We design TESLA to be secure against a powerful adversary with the following capabilities:

¸ Full control over the network. The adversary can eavesdrop, capture, drop, resend, delay, and alter packets.

¸ Access to a fast network with negligible delay.

¸ The adversary's computational resources may be very large, but not unbounded. In particular, this means that the adversary can perform efficient computations, such as computing a reasonable number of pseudo-random function applications and MACs with negligible delay. Nonetheless, the adversary cannot find the key of a pseudorandom function (or distinguish it from a random function) with non-negligible probability.

The security property of TESLA guarantees that the receiver never accepts M_i as an authentic message unless the sender really sent M_i. A scheme that provides this guarantee is called a secure broadcast authentication scheme.

Since TESLA requires the receiver to buffer packets before authentication, the receiver needs to protect itself from a potential denial-of-service (DOS) attack due to a flood of bogus packets.

2.2 Assumptions

TESLA makes the following assumptions in order to provide security:

1. The sender and the receiver must be be able to loosely synchronize. Specifically, each receiver must be able to compute an upper bound on the lag of the receiver clock relative to the sender clock. We denote this quantity by D_t. (That is, D_t = sender time - receiver time).
   We note that an upper bound on D_t can be easily obtained via a simple two-message exchange. (Such an exchange can be piggybacked on any session initiation protocol. Alternatively, standard protocols such as as NTP [16] can be used.
   (The synchronization assumption of TESLA is considerably weaker the synchronization requirements of authentication protocols based on timestamps. In those protocols, the participants are

assumed to have the same global time a-priori.)

2. TESLA MUST be bootstrapped at session set-up through a regular data authentication system. We recommend to use a digital signature algorithm for this purpose, in which case the receiver is REQUIRED to have an authentic copy of either the sender's public key certificate or a root key certificate in case of a PKI (public-key infrastructure).

3. TESLA uses cryptographic MAC and PRF (pseudo-random functions). These MUST be cryptographically secure. Further details on the instantiation of the MAC and PRF are in Section 4.2.

4. We would like to emphasize that the security of TESLA does NOT rely on any assumptions on network propagation delay.

3 The Basic TESLA Protocol

TESLA is described in several academic publications: A book on broadcast security [13], a journal paper [14], and two conference papers

[8,15]. Please refer to these publications for an in-depth treatment.

3.1 Sketch of protocol

We first outline the main ideas behind TESLA.

As we argue in the introduction, broadcast authentication requires a source of asymmetry. TESLA uses time for asymmetry. We first make sure that the sender and receivers are loosely time synchronized as described above. Next, the sender forms a one-way chain of keys, where each key in chain is associated with a time interval (say, a second). Here is the basic approach:

¸ The sender attaches a MAC to each packet. The MAC is computed over the contents of the packet. For each packet, the sender uses the current key from the one-way chain as a cryptographic key to compute the MAC.

¸ The sender discloses a key from the one-way chain after some pre-defined time delay. (e.g., the key used in time interval i is disclosed at time interval i+3.)

¸ Each receiver receives the packet. Each receiver knows the schedule for disclosing keys and, since it has an upper bound on the local time at the sender, it can check that the key used to compute the MAC was not yet disclosed by the sender. If so, then the receiver buffers the packet. Otherwise the packet is dropped. (Note that we do not know for sure whether a "late packet" is a bogus one or simply a delayed packet. We drop the packet since we are unable to authenticate it.)

¸ Each receiver checks that the disclosed key belongs to the hash-chain (by checking against previously released keys in the chain) and then checks the correctness of the MAC. If the MAC is correct, the receiver accepts the packet.

Note that one-way chains have the property that if intermediate values of the one-way chain are lost, they can be recomputed using subsequent values in the chain . So, even if some key disclosures are lost, a receiver can recover the corresponding keys and check the correctness of earlier packets.

We now describe the stages of the basic TESLA protocol in this order: sender setup, receiver bootstrap, sender transmission of authenticated broadcast messages, and receiver authentication of broadcast messages.

3.2 Sender Setup

The sender divides the time into uniform intervals of duration T_int. The sender assigns one key from the one-way chain to each time interval in sequence.

The sender determines the length N of the one-way chain K_0, K_1, ..., K_N, and this length limits the maximum transmission duration before a new one-way chain must be created. The sender picks a random value for K_N. Using a pseudo-random function (PRF) f, the sender constructs the one-way function F: F(k) = f_k(0). The rest of the chain is computed recursively using K_i = F(K_{i+1}). Note that this gives us K_i = F^{N-i}(K_N), so the receiver can compute any value in the key chain from K_N even if is does not have intermediate values.

The key K_i will be used to authenticate packets sent in time interval i.

3.3 Bootstrapping Receivers

Before a receiver can authenticate messages with TESLA, it needs to have:
* An upper bound D_t on the lag of its own clock with respect to the clock of the sender. (That is, if the local time reading is t, the current time reading at the sender is at most t+D_t.).
* The disclosure schedule of keys. (Note that this information is not essential. See details below.)
* One authenticated key of the one-way key chain. (Typically, this will be the last key in the chain, i.e. K_0, this key will be signed by the sender, and all receivers will verify the signature against the public key of the signer.

The sender sends the key disclosure schedule by transmitting the following information to the receivers over an authenticated channel (either via a digitally signed broadcast message, or over an authenticated unicast channel with each receiver):

¸ Time interval schedule: interval duration T_int, start time of interval i and index of interval i, length of one-way key chain.

¸ Key disclosure delay d (number of intervals).

¸ A commitment to the key chain K_i (i < j - d + 1, where j is the current interval index).

The receiver can perform the time synchronization and getting the authenticated TESLA parameters in a two-round message exchange, which we will describe in the technical TESLA document. Time synchronization can be performed as part of the registration protocol between member and sender.

3.3.1 Time Synchronization

Various approaches exist for time synchronization [16,17,18,19]. TESLA, however, only requires the receiver to know an upper bound on the delay of its local clock with respect to the receiver's clock, so a simple algorithm is sufficient. TESLA can be used with direct, indirect, and delayed synchronization as three default options.

The specific synchronization method will be part of each instantiation of TESLA, and needs to be described in the appropriate standards-track RFC.

For completeness we sketch a simple method for direct synchronization between the sender and a receiver:

* The receiver sends a (sync t_r) message to the sender and records its local time t_r.
* Upon receipt of the (sync t_r) message, the sender records its local time t_s and sends (synch, t_r,t_s) to the receiver.
* Upon receiving (synch,t_r,t_s), the receiver sets D_t = t_s - t_r + S, where S is an estimated bound on the clock drift throughout the duration of the session.

Note:

* Assuming that the messages are authentic (i.e., the message received the receiver was actually sent by the sender), and assuming that the clock drift is at most S, then at any point throughout the session we have 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.

* The exchange of sync messages needs to be authenticated. This can be done in a number of ways, for instance a secure NTP protocol, or in conjunction with a session set-up protocol.

3.4 Broadcasting Authenticated Messages

Each key in the one-way key chain corresponds to a time interval. Every time a sender broadcasts a message, it appends a MAC to the message, using the key corresponding to the current time interval. The key remains secret for the next d-1 intervals, so messages a sender broadcasts in interval j effectively disclose key K_j-d. We call d the key disclosure delay.

We do not want to use the same key multiple times in different cryptographic operations, that is, to use key K_j to derive the previous key of the one-way key chain K_{j-1}, and to use the same key K_j as the key to compute the MACs in time interval j may potentially lead to a cryptographic weakness. Using a pseudo-random function (PRF) f', we construct the one-way function F': F'(k) = f'_k(1). We use F' to derive the key to compute the MAC of messages in each interval. The sender derives the MAC key as follows: K'_i = F'(K_i). Figure 1 depicts the one-way key chain construction and MAC key derivation. To broadcast message M_j in interval i the sender constructs packet P_j = {M_j || i || MAC(K'_i,M_j) || K_{i-d}}, where || denotes concatenation.

```
                 F(K_i)     F(K_{i+1})      F(K_{i+2})
     K_{i-1} <------- K_i <--------- Ki+1 <-------   Ki+2

         |             |              |
         | F'(K_{i-1}) | F'(K_i)      | F'(K_{i+1})
         |             |              |
         V             V              V

        K'_{i-1}      K'_i          K'_{i+1}
```

Figure 1: At the top of the figure, we see the one-way key chain (derived using the one-way function F), and the derived MAC keys (derived using the one-way function F').

3.5 Authentication at Receiver

Once a sender discloses a key, we must assume that all parties might have access to that key. An adversary could create a bogus message and forge a MAC using the disclosed key. So whenever a packet arrives, the receiver must verify that the MAC is based on a safe key; a safe key is one that is still secret (only known by the sender). We define a safe packet or safe message to be one with a MAC that is computed with a safe key.

If the packet is not safe, the receiver must discard that packet, because the authenticity is not assured any more.

We now explain the TESLA authentication in more detail. When the receiver receives packet P_j sent in interval i, the receiver computes an upper bound on the sender's clock: t_j. To test whether the packet is safe, the receiver computes the highest interval x the sender could possibly be in, namely x = floor((t_j - T_0) / T_int). The receiver verifies that x < i + d (where i is the interval index), which implies that the sender is not yet in the interval during which it discloses the key K_i. If the condition fails then the receiver drops the packet.

The receiver cannot yet verify the authenticity of packets sent in interval i without key K_i. Instead, it adds the triplet (i, M_j, MAC(K'_i, M_j)) to a buffer, and verifies the authenticity after it learns K'_i.

What does a receiver do when it receives the disclosed key K_i? First, it checks whether it already knows K_i or a later key K_j (j>i). If K_i is the latest key received to date, the receiver checks the legitimacy of K_i by verifying, for some earlier key K_v (v<i) that K_v = F^{i-v}(K_i). The receiver then computes K'_i = F'(K_i) and verifies the authenticity of packets of interval i.

Using a disclosed key, we can calculate all previous disclosed keys, so even if packets are lost, we will still be able to verify buffered, safe packets from earlier time intervals. Thus, if i-v>1, the receiver can also verify the authenticity of the stored packets of intervals v+1 ... i-1.

Note that the security of TESLA does not rely on any assumptions on network propagation delay.

3.6 Determining the Key Disclosure Delay

An important TESLA parameter is the key disclosure delay d. Although the choice of the disclosure delay does not affect the security of the system, it is an important performance factor. A short disclosure delay will cause packets to lose their safety property, so receivers will discard them; but a long disclosure delay leads to a long authentication delay for receivers. We recommend choosing the disclo¡ sure delay as follows: in direct time synchronization let the RTT be a reasonable upper bound on the round trip time between the sender and the receiver; then choose d = ceil(RTT / T_int) + 1. Note that rounding up the quotient ensures that d >= 2. Also note that a disclosure delay of one time interval (d=1) does not work. Consider packets sent close to the boundary of the time interval: after the network propagation delay and the receiver time synchronization error, a receiver will need to discard the packet, because the sender will already be in the next time interval, when it discloses the corresponding key.

3.7 An alternative delay description method

The above description instructs the sender to include the time interval i in each packet. The receiver then uses i to determine the time at which the key authenticating the packet is disclosed. This method limits the the sender to a pre-determined schedule of disclosing keys.

Alternatively, the sender may directly include in each packet the time t_p at which it is going to disclose the key for this packet. This way, the receiver does not need to know the duration of intervals or the delay factor d. All the receiver needs to know is the bound D_t on the clock skew and T_0, the sender's local time at the initiation of the session. Then the receiver records the local time T when the packet has arrived, and verifies that

$$T <= T\_0 + D\_t + t\_p.$$

Else the packet is dropped.

Another advantage of this method is that the sender is able to change the duration of intervals and the key disclosure delay dynamically throughout the session.

3.8 Some extensions

Let us mention two salient extensions of the basic TESLA scheme. A first extension allows having multiple TESLA authentication chains for a single stream, where each chain uses a different delay for disclosing the keys. This extension is typically used to deal with heterogeneous network delays within a single multicast transmission. A second extension allows having most of the buffering of packets at the sender side (rather than at the receiver side). Both extensions are described in [15].

4 Layer placement

The TESLA authentication can be performed at any layer in the networking stack. Three natural places are in the network, transport, or the application layer. We list some considerations regarding the choice of layer:

¸ Performing TESLA in the network layer has the advantage that the transport or application layer only receives authenticated data, potentially aiding a reliability protocol and preventing denial of service attacks. (Indeed, reliable multicast tools based on forward error correction are highly susceptible to denial of

service due to bogus packets.)

¸ Performing TESLA in either the transport or the application layer has the advantage that the network layer remains unchanged; but it has the drawback that packets are obtained by the application layer only after being processed by the transport layer. Consequently, if TCP is used then this may introduce additional and unpredictable delays on top of the unavoidable network delays. (However, if UDP is used then this is not a problem.)

5. Security Considerations

See the academic publications on TESLA [8,14,20] for several security analyses. Regarding the security of implementations, by far the most delicate point is the verification of the timing conditions. Care should be taken to to make sure that:
(a) The value bound D_t on the clock skew is calculated according to the spec at session set-up.
(b) The receiver records the arrival time of the packet as soon as possible after the packet's arrival, and computes the safety condition correctly.

6 Acknowledgments

We would like to thank Mike Luby for his feedback and support.

7 Bibliography

[1] T. Dierks and C. Allen, "The TLS protocol version 1.0." Internet Request for Comments RFC 2246, January 1999. Proposed standard.

[2] Ipsec, "IP Security Protocol, IETF working group." http://www.ietf.org/html.charters/ipsec-charter.html.

[3] D. Boneh, G. Durfee, and M. Franklin, "Lower bounds for multicast message authentication," in Advances in Cryptology -- EUROCRYPT '2001 (B. Pfitzmann, ed.), vol. 2045 of Lecture Notes in Computer Science , (Innsbruck, Austria), pp. 434--450, Springer-Verlag, Berlin Germany,

2001.

[4] R. Gennaro and P. Rohatgi, "How to Sign Digital Streams," tech. rep., IBM T.J.Watson Research Center, 1997.

[5] P. Rohatgi, "A compact and fast hybrid signature scheme for mul¡ ticast packet authentication," in 6th ACM Conference on Computer and Communications Security , November 1999.

[6] P. Rohatgi, "A hybrid signature scheme for multicast source authentication," Internet Draft, Internet Engineering Task Force, June 1999. Work in progress.

[7] C. K. Wong and S. S. Lam, "Digital signatures for flows and mul¡ ticasts," in Proc. IEEE ICNP `98 , 1998.

[8] A. Perrig, R. Canetti, J. Tygar, and D. X. Song, "Efficient authentication and signing of multicast streams over lossy channels," in IEEE Symposium on Security and Privacy , May 2000.

[9] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: A taxonomy and some efficient construc¡ tions," in Infocom '99 , 1999.

[10] S. Cheung, "An efficient message authentication scheme for link state routing," in 13th Annual Computer Security Applications Confer¡ ence , 1997.

[11] F. Bergadano, D. Cavagnino, and B. Crispo, "Chained stream authentication," in Selected Areas in Cryptography 2000 , (Waterloo, Canada), August 2000. A talk describing this scheme was given at IBM Watson in August 1998.

[12] F. Bergadano, D. Cavalino, and B. Crispo, "Individual single source authentication on the mbone," in ICME 2000 , Aug 2000. A talk containing this work was given at IBM Watson, August 1998.

[13] A. Perrig and J. D. Tygar, Secure Broadcast Communication in Wired and Wireless Networks Kluwer Academic Publishers, Oct. 2002. ISBN 0792376501.

[14] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The tesla broadcast authentication protocol," RSA CryptoBytes , vol. 5, no.

Summer, 2002.

[15] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in Network and Dis¡ tributed System Security Symposium, NDSS '01 , pp. 35--46, February 2001.

[16] D. L. Mills, "Network Time Protocol (Version 3) Specification, Implementation and Analysis." Internet Request for Comments, March 1992. RFC 1305.

[17] B. Simons, J. Lundelius-Welch, and N. Lynch, "An overview of clock synchronization," in Fault-Tolerant Distributed Computing (B. Simons and A. Spector, eds.), no. 448 in LNCS, pp. 84--96, Springer-Verlag, Berlin Germany, 1990.

[18] D. Mills, "Improved algorithms for synchronizing computer net¡ work clocks," in Proceedings of the conference on Communications architectures, protocols and applications, SIGCOMM 94 , (London, England), pp. 317--327, 1994.

[19] L. Lamport and P. Melliar-Smith, "Synchronizing clocks in the presence of faults," J. ACM , vol. 32, no. 1, pp. 52--78, 1985.

[20] Philippa Broadfoot and Gavin Lowe, "Analysing a Stream Authentication Protocol using Model Checking. In Proceedings of the 7th European Symposium on Research in Computer Security (ESORICS), 2002.

A Author Contact Information

Adrian Perrig
ECE Department
Carnegie Mellon University
Pittsburgh, PA
US
perrig@ece.cmu.edu

Ran Canetti
IBM Research
30 Saw Mill River Rd

Hawthorne, NY 10532
US
canetti@watson.ibm.com

Dawn Song
CS Department
Carnegie Mellon University
Pittsburgh, PA
US
dawnsong@cmu.edu

Doug Tygar
UC Berkeley

102 South Hall, 4600
Berkeley, CA 94720-4600
US
tygar@cs.berkeley.edu

Bob Briscoe
BT Research
B54/74, BT Labs
Martlesham Heath
Ipswich, IP5 3RE
UK
bob.briscoe@bt.com

B Full Copyright Statement

Copyright (C) The Internet Society (2000). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this doc¡ ument itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of develop¡

ing Internet standards in which case the procedures for copyrights defined in the Internet languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MER¡ CHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."