Internet Engineering Task Force IETF MSEC
Internet Draft Perrig, Canetti, Song, Tygar, Briscoe
draft-ietf-msec-tesla-intro-01.txt UC Berkeley/Digital Fountain/IBM/BT
27 October 2002
Expires: 27 April 2002

TESLA: Multicast Source Authentication Transform Introduction

STATUS OF THIS MEMO

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 authen tication mechanism for multicast or broadcast data streams. This doc ument provides an algorithmic description of the scheme for informa tional purposes, and in particular, it is intended to assist in writ ing 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 toler ance, high efficiency, no per-receiver state at the sender. The prob lem 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 authen tication 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 com municated 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 set ting. Consider a sender that broadcasts authentic data to mutually untrusted 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 mech anism 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 signa tures 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 immedi ately 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 draft presents the Timed Efficient Stream Loss-tolerant Authen tication protocol (TESLA). TESLA uses mainly symmetric cryptography,

and uses time delayed key disclosure to achieve the required asymme try 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].

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 both in the network 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, cap

ture, 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 neg ligible delay. Nonetheless, the adversary cannot find the key of a pseudorandom function (or distinguish it from a random func tion) 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 broad cast authentication scheme.

Since TESLA requires the receiver to buffer packets before authenti cation, 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 loosely time synchronized. Loosely time synchronized means that the synchronization does not need to be precise, but the receiver MUST know an upper bound on the dispersion (the maximum clock offset). For the purposes of this draft, we assume that the receiver knows the maximum clock offset between its clock and the sender's clock, which we denote with D_t. We stress that the sender and receiver's clock do not need to be synchronized a-priori. Instead, the receiver can easily achieve the required synchro nization through a two-round message exchange with the sender. (This stands in contrast with 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 sig

nature 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 func tions). 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 Notation

To denote the subscript or an index of a variable, we use the under score 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).

4 The Basic TESLA Protocol

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

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

4.1 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 assume that the sender and receivers are all loosely time synchronized -- up to some D_t value, all parties agree on the current time. The sender forms a one-way chain, where each such value 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 value from the one-way chain as a cryptographic key to compute the MAC.

- Each receiver receives the packet. Each receiver knows the sched ule for disclosing keys and, since the clocks are loosely syn chronized, can check that the key used to compute the MAC is still secret by determining that the sender could not have yet reached the time for disclosing it. If the MAC key is still secret, then the receiver buffers the packet.

- According to a schedule, the sender discloses the key from the one-way chain.

- Each receiver checks that the disclosed key is correct (using previously released keys) 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 val ues of the one-way chain are lost, they can be recomputed using the following values. So, even if some key disclosures are lost, a receiver can recover the key chain and check the correctness of ear lier packets.

The sender distributes a stream of messages {M_i}, and the sender sends each message M_i in a network packet P_i along with authentica tion information. The broadcast channel may be lossy, but in many broadcast applications the sender does not retransmit lost packets. Despite packet loss, each receiver needs to authenticate every mes sage it receives.

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

messages.

4.2 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 inter val 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 inter val i.

4.3 Bootstrapping Receivers

Before a receiver can authenticate messages with TESLA, it needs to be loosely time synchronized with the sender, know the disclosure schedule of keys, and receive an authenticated key of the one-way key chain.

Various approaches exist for time synchronization [16,17,18,19]. TESLA, however, only requires loose time synchronization between the sender and the receivers, so a simple algorithm is sufficient. The time synchronization property that TESLA requires is that each receiver can place an upper bound of the senders local time. TESLA offers direct, indirect, and delayed synchronization as three default options, which we will describe in the TESLA technical draft.

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

- Time interval schedule: interval duration T_int, start time and index of interval i, length of one-way key chain.
- Key disclosure delay d (number of intervals).
- A key 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 draft. Time synchronization can be performed as part of the registration protocol between member and sender.

4.4 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 cryp tographic 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 || MAC(K'_i,M_j) || K_{i-d}}, where || denotes concatenation.

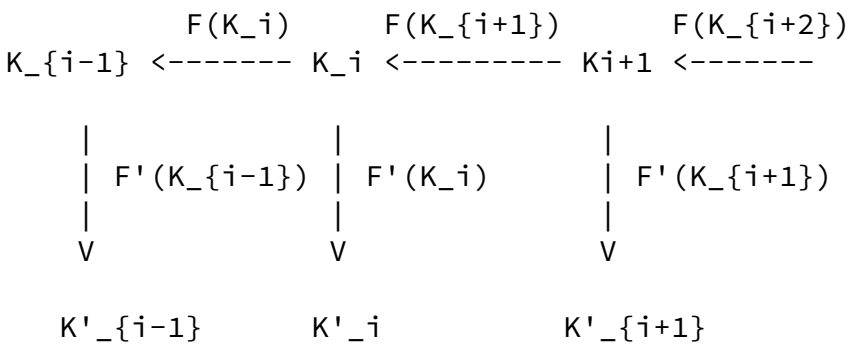

Figure 1: At the top of the figure, we see the one-way key chain (derived using the one-way function F), and the derived MAC keys (derived using the one-way function F').

4.5 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 com putes 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.

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.

4.6 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 loose 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 dis closure 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 cor responding key.

4.7 Some extenstions

Let us mention two salient extenstions 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 heterogenous network delays withing 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].

5 Layer placement

The TESLA authentication can be performed at any layer in the net working stack. The two logical places are in the network 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 ser vice due to bogus packets.)

- Performing TESLA in 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.)

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, Eng land), 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.

A Author Contact Information

Adrian Perrig
UC Berkeley / Digital Fountain
102 South Hall
Berkeley, CA 94720
US
perrig@cs.berkeley.edu

Ran Canetti
IBM Research
30 Saw Mill River Rd
Hawthorne, NY 10532
US
canetti@watson.ibm.com

Dawn Song
UC Berkeley
387 Soda Hall, 1776
Berkeley, CA 94720-1776
US
dawnsong@cs.berkeley.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