Internet Engineering Task Force IETF MSEC
Internet Draft Perrig, Canetti, Song, Tygar, Briscoe
draft-ietf-msec-tesla-intro-03.txt CMU / IBM / CMU / UC Berkeley / BT
August 2004
Expires: 1 February 2005

TESLA: Multicast Source Authentication Transform Introduction

STATUS OF THIS MEMO

Abstract

Data authentication is an important component for many broadcast applications, for example audio and video Internet broadcasts, or data distribution by satellite. This document introduces TESLA, short for Timed Efficient Stream Loss-tolerant Authentication, a secure source authentication mechanism for multicast or broadcast data streams. This document provides an algorithmic description of TESLA for informational purposes, and in particular, 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, efficiency, and 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 receivers want to authenticate each packet they do receive.

TESLA provides authentication of individual data packets, regardless of 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

In multicast, a single packet can reach millions of receivers. This unfortunately also introduces the danger that an attacker can potentially also reach millions of receivers with a malicious packet. Through source authentication, receivers can ensure that a received multicast packet originates from the correct source.

In unicast communication, we can achieve data authentication through a simple 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, e.g., 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, a stronger property than authentication. However, digital signatures have a high cost: they have a high computation overhead for both the sender and the receiver, and most signatures also have a high bandwidth 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. 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., function F with the argument x executed i times is F^i(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. Certain applications intolerant of delay may be willing to process packets in parallel to being buffered awaiting authentication, as long as roll-back is possible if packets are later found to be

unauthenticated. For instance, an interactive video may play-out packets still awaiting authentication, but if they are later found to be unauthenticated, it could stop further play-out and warn the viewer that the last x msec were unauthenticated and should be ignored. However, in the remainder of this document, for brevity, we will assume packets are not processed in parallel to buffering.

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 expects 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 secure session initiation protocol. Alternatively, standard protocols such as as NTP [16] can be used.

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). Alternatively, this initialization step can be done using any secure session initiation protocol.

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 Protocol sketch

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 due to inability to authenticate. 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. (Ofcourse, an implementation may choose to not drop packets and use them unauthenticated.)

- 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.

Jakobsson [21], and Coppersmith and Jakobsson [22] present a storage and computation efficient mechanism for one-way chains. For a chain of length N, storage is about log(N) elements, and the computation overhead to reconstruct each element is also about log(N).

The sender determines the duration of a time interval, T_int, and the key disclodure delay, d. (T_int is measured in time units, say millieconds, and d is measured in number of time intervals. That is, a key that is used for time interval i will be desclosed in time interval i+d.) It is stressed that the scheme remains secure for any values of T_int and d. Still, correct choice of T_int and d is crucial for the usability of the scheme. The choice is influenced by the estimated network delay, the length of the transmission, and the tolerable delay at the receiver. T_int that is too short will cause the keys to run out too soon. T_int that is too long will cause excessive delay in authentication for some of the packets (those that were sent at the beginning of a time period). Delay d that is too short will cause to many packets to be unverifiable by the receiver. Delay d that is too long will cause excessive delay in authentication.

If the sender estimates that the average network delay is m milliseconds, and the sender expects to send a packet every n milliseconds, then a reasonable value for T_int is max(n,m) and a reasonable value for d is ceil(2m/T_int). If the application can tolerate higher authentication delay then T_int can be made appropriately larger.

Finally, the sender needs to allow each receiver to synchronize its time with the sender. See more details on how this can be done in section 3.3. (It is stressed that estimating the network delay is a separate task than the time synchronization between the sender and the receivers.)

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.).

- 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 disclosure schedule of keys:

    - T_int, the interval duration.
    - T_0, the start time of interval 1.
    - N, the length of the one-way key chain.
    - d, the key disclosure delay d (in number of intervals).

The receiver can perform the time synchronization and getting the authenticated TESLA parameters in a two-round message exchange, as described below. We stress again that time synchronization can be performed as part of the registration protocol between the receiver and sender, or between the receiver and a group controller.

3.3.1 Time Synchronization

Various approaches exist for time synchronization [16,17,18,19]. TESLA only requires the receiver to know an upper bound on the delay of its local clock with respect to the sender'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.

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.

For indirect time synchronization (eg, synchronization via a group controller), the sender and the controller engage in a protocol for finding the value D^0_t between the sender and the controller. Next, each receiver R interacts with the group controller (say, when registering to the group) and finds the value D^R_t between the group controller and R. The overall value of D_t within R is set to the sum D_t = D^R_t + D^0_t.

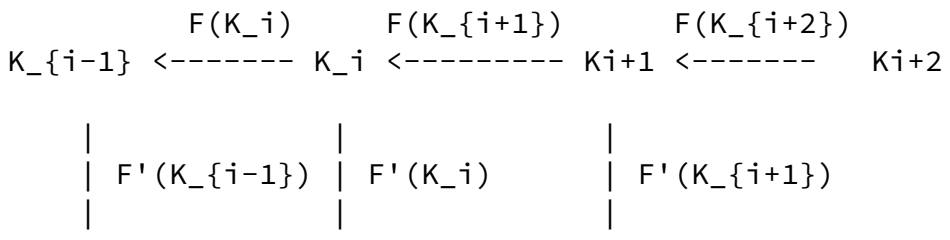
3.4 Broadcasting Authenticated Messages

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 the packet

$$P_j = \{M_j \,||\, i \,||\, MAC(K'_i, M_j) \,||\, K_{i-d}\}$$

where || denotes concatenation.



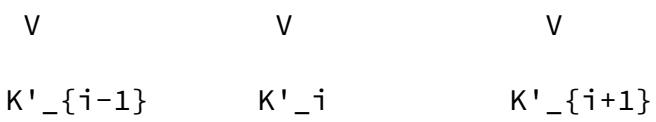


Figure 1: At the top of the figure, we see the one-way key chain (derived using the one-way function F), and the derived MAC keys (derived using the one-way function F').

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 which carries an interval index i, the receiver first records local time T in which the packet arrived. It then computes an upper bound t_j on the sender's clock at the time where the packet arrived: t_j = T + D_t. To test whether the packet is safe, the receiver then 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 considers the packet unauthenticated.

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 with j>i. Then 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.

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 determining the disclosure 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.

We stress that the security of TESLA does not rely on any assumptions on network propagation delay: If the delay is longer than expected then authentic packets may be considered unauthenticated. Still, no unauthentic packet will be accepted as authentic.

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 considered aunauthenticated.

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. It is stressed, however, that the interval index i must still be included in the packet, to allow the receiver to know which key K_i should be used to verify the packet.

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].

The requirement in TESLA to receive a key in a later packet for authentication prevents a receiver from authenticating the last part of a message. Thus, to enable authentication of the last part of a message or of the last message before a transmission suspension, the sender needs to send an empty message with the key to enable authentication.

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.)

- It should be kept in mind that, since TESLA relies upon timing of packets, deploying TESLA on top of a protocol or layer which aggressively buffers packets and hides the true packet arrival time (e.g., TCP) will significantly reduce the perormance of TESLA.

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.

It is important to note that, unless appropriate measures are taken, TESLA increases the susceptibility of receivers to denial of service (DoS) attacks. Specifically, an attacker can flood a victim receiver with bogus packets that need to be buffered for delayed authentication. There are several measures against such attacks. Here we mention some of the known mechanisms to reduce this susceptibility:

- Limit the size of buffers at the receiver side.

- Add an external MAC that is computed using either (a) a key that is constant for the session and known to all receivers, or (b) the key that is disclosed in the current packet. Both mechanisms prevent simple flooding attacks but are still susceptible to more sophisticated attacks that have some "inside" information on the session.

- Move the buffering of packets to the sender side, and allow receivers to authenticate packets immediately upon receipt. This method is described in [14].

Finally, in common with all authentication schemes, if verification is located separately from the ultimate destination application (e.g. an IPSec tunnel end point), a trusted channel must be present between verification and the application. For instance, the interface between the verifier and the application might simply assume that packets received by the application must have been verified by the verifier (because otherwise they would have been dropped). The application is then vulnerable to reception of packets that have managed to bypass the verifier.

6 IP Statement

The authors are not aware of any patents that encumber the free use of TESLA.

7 Acknowledgments

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

8 References

All references are informative.

[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 multicast 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 multicasts," 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 constructions," in Infocom '99 , 1999.

[10] S. Cheung, "An efficient message authentication scheme for link state routing," in 13th Annual Computer Security Applications Conference , 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 Distributed 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 network 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.

[21] M. Jakobsson, "Fractal hash sequence representation and traver¡sal." Cryptology ePrint Archive, http://eprint.iacr.org/2002/001/, Jan. 2002.

[22] D. Coppersmith and M. Jakobsson, "Almost optimal hash sequence traversal," in Proceedings of the Sixth International Financial Cryp¡tography Conference (FC '02) , March 2002.

A Author Contact Information

Adrian Perrig
ECE Department
Carnegie Mellon University
Pittsburgh, PA 15218
US
perrig@cmu.edu

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

Dawn Song
ECE Department
Carnegie Mellon University
Pittsburgh, PA 15218
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
B54/77, BT Labs
Martlesham Heath
Ipswich, IP5 3RE
UK
bob.briscoe@bt.com

B Full Copyright Statement