Reducing Delay and Enhancing DoS Resistance in Multicast Authentication Through Multigrade Security

Qing Li, *Student Member, IEEE*, and Wade Trappe, *Member, IEEE*

***Abstract*—Many techniques for multicast authentication employ the principle of delayed key disclosure. These methods introduce delay in authentication, employ receiver-side buffers, and are susceptible to denial-of-service (DoS) attacks. Delayed key disclosure schemes have a binary concept of authentication and do not incorporate any notion of partial trust. This paper introduces staggered timed efficient stream loss-tolerant authentication (TESLA), a method for achieving multigrade authentication in multicast scenarios that reduces the delay needed to filter forged multicast packets and, consequently, mitigates the effects of DoS attacks. Staggered TESLA involves modifications to the popular multicast authentication scheme, TESLA, by incorporating the notion of multilevel trust through the use of multiple, staggered authentication keys in creating message authentication codes (MACs) for a multicast packet. We provide guidelines for determining the appropriate buffer size, and show that the use of multiple MACs and, hence, multiple grades of authentication, allows the receiver to flush forged packets quicker than in conventional TESLA. As a result, staggered TESLA provides an advantage against DoS attacks compared to conventional TESLA. We then examine two new strategies for reducing the time needed for complete authentication. In the first strategy, the multicast source uses assurance of the trustworthiness of entities in a neighborhood of the source, in conjunction with the multigrade authentication provided by staggered TESLA. The second strategy achieves reduced delay by introducing additional key distributors in the network.**

***Index Terms*—Denial-of-service (DoS) attacks, forge-capable area, message authentication code (MAC), multigrade source authentication, queueing theory, timed efficient stream loss-tolerant authentication (TESLA), trust.**

I. Introduction

MULTICAST will play a significant role in the next generation of networks as many services, such as pay-per-view media broadcasts and the delivery of network control messages, will rely upon group communication [1]. One security service that has been difficult to provide for multicast is authentication. Existing solutions are either resource intensive, or introduce a significant delay in authentication. A consequence of the delay overhead associated with many multicast authentication schemes is that they rely on receiver-side buffers and are therefore susceptible to denial-of-service (DoS) attacks targeted at filling a receiver's buffer with false packets. Therefore, authentication strategies that allow for less delay and more efficient utilization of buffer resources are desirable.

One explanation for the inefficiency associated with multicast authentication stems from the underlying conceptual formulation of authentication. Authentication is about trust, and in the context of traditional network security services, trust is a binary concept. A binary formulation of trust is a deviation from our natural, social understanding of trust, where the confidence we place in others is not a black-and-white concept, but rather broken down into many shades of gray.

In this paper, our objective is to present strategies that reduce the delay associated with multicast authentication, make more efficient usage of receiver-side buffers, make delayed key disclosure authentication more resilient to buffer overflow denial of service attacks, and allow for multiple levels of trust in authentication. Throughout this paper, we will focus our discussion on the popular multicast authentication scheme timed efficient stream loss-tolerant authentication (TESLA), though our techniques can apply to other authentication methods based upon the delayed key disclosure principle. Similar to other schemes based upon delayed key disclosure, TESLA is susceptible to DoS attacks and is not well suited for delay-sensitive applications.

At the heart of our approach is a modification to TESLA, which we call staggered TESLA, that employs several message authentication codes (MACs) that correspond to authentication keys that are staggered in time. staggered MACs provide notions of partial authentication and allow for forged packets to be more readily removed from the buffer, thereby improving usage of the receiver's buffer. A benefit of partial authentication is that one may define security policies that allow for partially authenticated packets to pass through the buffer and, thus, packets will remain in the buffer for a shorter duration. In many scenarios, accepting partially authenticated packets is unacceptable and, therefore, we present two further techniques that may be used to reduce the delay needed for full authentication. The first strategy requires that the source has a guarantee that there are no adversaries within a certain network distance of the source. By having a guarantee of proximity protection, partially authenticated packets may be accepted as fully authentic. The second strategy for reducing full authentication delay that we present involves replicating the key distribution functionality within the network, and having a set of distributed key distributors transmits the key seeds. A benefit of all of these strategies is that they mitigate the threat of a buffer overflow DoS attack since an adversary must conduct a DoS attack at a higher attack rate.

Manuscript received May 3, 2005; revised February 2, 2006. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Klara Nahrstedt.

The authors are with the Wireless Information Network Laboratory (WINLAB), Rutgers University, Piscataway, NJ 08854 (e-mail: qingli@winlab.rutgers.edu; trappe@winlab.rutgers.edu).

Digital Object Identifier 10.1109/TIFS.2006.873599

The rest of the paper is organized as follows. In Section II, we review the related works in multicast source authentication, and give a brief overview of the conventional TESLA scheme. We explore partial trust and use it in Section III, where we describe the staggered TESLA scheme. The security requirements needed to reduce full authentication delay will be discussed in Section IV. We derive theoretical guidelines for buffer requirements and discuss the tradeoffs involved in staggered TESLA in Section V. We support the theoretical analysis by conducting simulations and present the results of the simulations in Section VI. Finally, Section VII concludes the paper.

II. Background Literature and TESLA

A. Related Work

Source authentication enables receivers to verify that the received data originated from the claimed source and was not modified. Source authentication in point-to-point communications can be solved by asymmetric cryptography. Asymmetric cryptography, however, consumes significant communication and computational resources that cannot be supplied by resource-limited devices. Source authentication can also be accomplished through symmetric cryptography by appending MACs to each packet. The problem of authenticating multicast is more complex than the unicast case when there are untrusted receivers in the multicast group. Simply applying MACs does not provide source authentication in multicast. Adversarial group members, who share the same secret key as benign group members, can easily create packets with MACs using this shared key. Since all users share the same key, the receivers cannot resolve the source of the packets.

Although digital signatures [2] can be applied to multicast authentication, they have prohibitive computational and communication overhead. Gennaro [3] and Wong [4] proposed schemes to mitigate communication overhead by amortizing a single signature across several packets. Rohatgi [5] introduced an improved approach that employs k-time signature schemes and has less delay. Another signature amortization scheme is based on an information dispersal algorithm that can tolerate a certain amount of packet loss [6], [7]. Recent efforts on signature amortization for multicast authentication have involved distillation codes and have focused on resistance to DoS attacks [8]. Another work along these lines was presented by Lysyanskaya *et al.* [9] in which a multicast authentication scheme based on a combination of digital signatures, hashes, and error-correction codes is presented.

Multicast-source authentication based on symmetric cryptography has attracted intense research. Canetti presented a solution to multicast source authentication based on verifying l different MACs using l different keys for each message [10]. Unlike the method proposed in this paper, the multiple MACs in [10] are calculated using independent keys that are not temporally linked. Further, their protocol is based on the assumption that no coalition of w bad receivers can forge packets for a specific receiver, but fails in the presence of a coalition of more than w users. Perrig constructed a signature scheme using one-way functions without trapdoors for broadcast authentication [11]. Xu and Sandhu [12] proposed two hop-by-hop authentication schemes suitable for Internet multicasting that use the multicast tree and are immune to DoS attacks. A consequence of their hop-by-hop assumption is that intermediate routers are required to be trusted and secure.

Another popular approach uses delayed key disclosure. Delayed key disclosure was first introduced by Cheung [13] to achieve authentication for link-state routing, and was used in the Guy Fawkes protocol to provide nonrepudiation in unicast communication in [14]. Chained stream authentication [15], [16] and FLAMeS [17] used similar ideas for source authentication in multicast. In delayed key disclosure, the sender keeps the key secret for some intervals of time after sending the data. The receivers buffer the packets since they do not have the authentication key. A short time later, the sender discloses the key and the receivers are able to perform authentication. Using the delayed key disclosure introduces two new issues. The first issue is the buffer requirements at the receiver. Because the receiver needs to buffer the received packets before it can authenticate them, an adversary can launch a DoS attack and fill up the receiver's buffer with bogus traffic. The receiver will have to drop packets due to a lack of buffer space. Second, many applications are sensitive to delay, and reducing delay is critical for achieving a desirable quality of service. As we shall discuss later in this paper, reducing delay in delayed key disclosure schemes can be accomplished by either employing partial authentication or suitable assumptions about the application's security policy or the source's network neighborhood.

B. TESLA Overview

Among the many existing schemes employing the delayed disclosure principle, the TESLA [18]–[21] scheme is one of the most popular. We shall use the TESLA scheme as the basis for our discussions. TESLA is based on initial loose time synchronization between the sender and the receivers. TESLA divides time into intervals of equal duration, and each time slot n is assigned a corresponding key K_n. For each packet generated during time interval n, the sender appends a MAC that is created using the authentication key K_n. Each receiver buffers the packets without being able to authenticate them, until the sender discloses the key K_n by broadcasting the corresponding key-seed s_n. Once s_n is disclosed, anyone with s_n can calculate K_n and can pretend to be the sender by forging MACs. Thus, the use of K_n for creating MACs is limited to time interval n, and future time intervals use future keys. Further, s_n is not disclosed until d time slots later, where d is governed by an estimate of the maximum network delay for all recipients.

The keys K_n are derived from s_n using a publicly available one-way function F', while the s_n are related to each other via a reverse-time chain of one-way functions. To create the chain of keyseeds, the sender chooses a terminal seed s_l, and generates s_{l-1} using a one-way function F. The remaining seeds $\{s_0, s_1, \cdots, s_l\}$ are derived via $s_l \xrightarrow{F} s_{l-1} \xrightarrow{F} s_{l-2} \xrightarrow{F} \ldots \xrightarrow{F} s_1 \xrightarrow{F} s_0$. The sender uses the seedchain in the opposite direction (starting with seed s_0) to derive the TESLA keys by applying the one-way function F' via $s_n \xrightarrow{F'} K_n$.

When a user receives a packet, he or she first checks whether the packet is fresh (i.e., it was sent in a timeslot whose

TESLA-key has not been disclosed) or dated. The receiver discards all dated packets and buffers only the fresh ones. Once the user receives a TESLA seed s_n, he or she checks $F(s_n) = s_{n-1}$ to be sure of s_n's authenticity. He or she derives K_n by $K_n = F'(s_n)$, and authenticates the packets that were sent in timeslot n. The conditions needed for the verification of the safe keys are collectively referred to as the security condition for TESLA. The use of chained key seeds also provides resilience to packet loss. If intermediate key seeds are not received, then a future key seed may be authenticated by applying the one-way function F multiple times. The one-way function chain additionally allows for the determination of the packet's time of creation.

Several modifications are proposed in [19], where receiver buffering is traded off at the expense of source buffering as well as a scheme, called concurrent TESLA, that is suitable for different receiver delays. The multiple MACs used in concurrent TESLA correspond to multiple instantiations of the basic TESLA protocol, where each instantiation employs a different disclosure delay. This differs from the use of multiple MACs that we propose in staggered TESLA, where our multiple MACs correspond to a single instance of TESLA using a single, assumed disclosure delay.

C. Examination of Trust in TESLA

We now examine the notion of trust in TESLA, and how it can be modified to achieve partial trust. In TESLA, the seed s_i for the authentication key K_i is released at a later time interval $i + d$, where d is a value greater than the maximum number of time intervals needed for a message to travel from the source to all of the receivers. As a result, the total time that a packet will occupy the receiver's buffer is approximately d intervals. Let us now consider the objective of the adversary. The adversary seeks to replace the content of the packets and makes them pass the authentication check at the receiver. Thus, the adversary needs to know the key K_i in order to successfully forge packets sent during interval i. Since the seed for key K_i is released during time interval $i+d$, the receivers do not accept any packets that claim to have been created during interval i after the start of time interval $i + d$. Thus, adversaries are unable to forge MACs for interval i.

Now, let us consider what would happen if we send out the seed s_i earlier than in conventional TESLA. If s_i is sent at time interval $i + t$ instead of $i + d$, where $t < d$, then the receivers can authenticate packets sent in interval i when they receive the first packet sent in interval $i+t$. Consequently, the receivers can perform authentication sooner than they would have in conventional TESLA, and can thus remove the packets from the buffer earlier than in conventional TESLA. On the other hand, because the seeds are released earlier, some adversaries can take advantage of this and forge packets with valid MACs. Thus, authenticated packets cannot be classified as "fully trusted" and may be viewed as partially authenticated.

Our work is based upon this concept of partial authentication, and we therefore need to identify which entities are capable of forging packets with valid MACs at a specific time. Consider Fig. 1, where S corresponds to the sender, R depicts the receiver, and A_1 and A_2 are two adversaries at different network delay

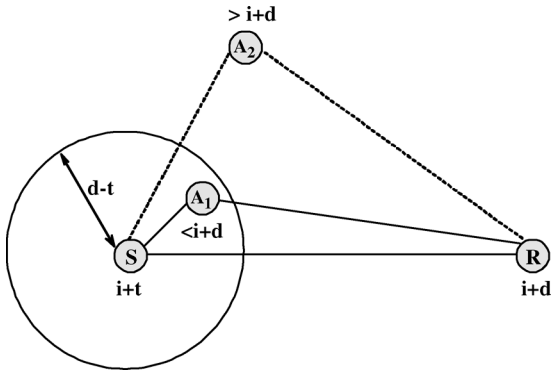


Fig. 1. Network diagram depicting relative network distances for the source S and receiver R for a single packet transmission. In TESLA, the network has a maximum network delay of d. For a single packet containing a key seed, adversary A_1 is within a radius of $d - t$ from the source, while adversary A_2 is beyond a radius $d - t$.

distances relative to the source and the receiver. A_1 is within a distance of $d-t$ from the source, while A_2 is a distance greater than $d-t$ from S. The distance in the figure represents the relative network time delay between entities for the transmission of a single-key seed packet. These network delay positions might change from packet to packet or interval to interval based upon network conditions. For simplicity, we assume that both adversaries do not require any time to process and forge packets. Additionally, we assume that the link between an adversary A_j and the receiver R is a very-high-speed link (perhaps dedicated for the purpose of performing a DoS) and, thus, for discussion, we consider the adversary-receiver links as 0-delay links. If the key is released in interval $i + t$, then all adversaries within a $(d - t)$ radius of the source, such as A_1, will receive the key before the start of interval $i + d$. Since the adversaries have 0-forge time and 0-delay links to the receiver, the receiver will receive packets forged by A_1 before the beginning of time interval $i+d$. The receiver will perceive that these packets obey the security condition, and put them into the buffer. Adversaries outside the circle of radius $(d - t)$, such as A_2 will receive the key after the start of interval $i+d$. Hence, they cannot forge packets with valid MACs. Therefore, exactly those adversaries that lie within a radius of $(d-t)$ delay from the source can successfully forge packets, and belong to the forge-capable area for that key seed. Hence, if we release the key seed at interval $i + t$, any packet from interval i that passes the authentication check can only be declared as partially trusted since there is a network area capable of forging that packet.

III. Staggered TESLA: Multigrade Multicast Authentication

We now use the idea of releasing key seeds earlier than in conventional TESLA to achieve multigrade multicast authentication. We begin the same way as TESLA by splitting the time into equal length intervals and assigning a seed s_n to time interval n. The authentication key K_n for interval n is derived from s_n using a publicly available one-way function F'. Our motivating observation is that many TESLA authentication keys will not be known by an adversary at an arbitrary location at an arbitrary time since it takes time for released keys to arrive at an

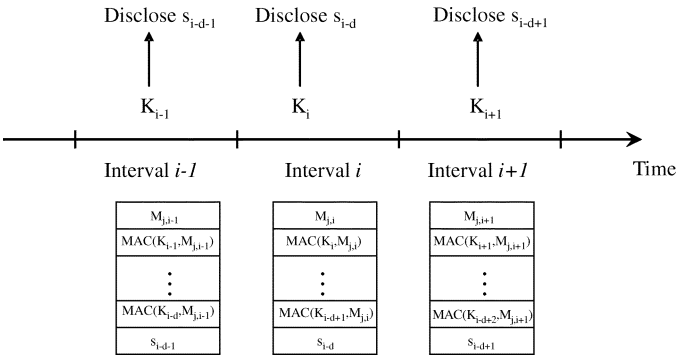


Fig. 2. Format of the jth packet sent during interval $i-1$, i, and $i+1$ in staggered TESLA. There are d MACs attached to each packet, compared with only one MAC attached to each packet in TESLA.

adversary. Thus, when forging packets from interval i, which corresponds to key K_i being used to create MACs, adversaries might also not know K_{i-1} or K_{i-2}. Therefore, if we use more than just K_i to construct MACs during time interval i, such as using K_{i-1} and K_{i-2}, many potential adversaries will not be able to forge the MACs constructed using K_{i-1} or K_{i-2}, much less than the MAC that used K_i. The idea of using MACs from successive TESLA keys leads to a scheme, which we call staggered TESLA.

A. Format of the Packet

In TESLA, a MAC computed by the authentication key corresponding to the current interval is attached to each packet. Let $M_{j,i}$ denote the jth message sent in interval i, K_i to be the authentication key used in interval i, and d to be the key disclosure delay in units of intervals. The source will disclose the key seed s_{i-d} in interval i. The receiver may use the seed to determine what time interval a packet was sent. The format of the jth packet sent in interval i is $\{M_{j,i}, \mathrm{MAC}(\mathrm{K_i}, \mathrm{M_{j,i}}), \mathrm{s_{i-d}}\}$.

In staggered TESLA, we attach additional MACs made from previous TESLA keys to each packet. Because the seed s_{i-d} is released in interval i, attaching a MAC computed using key K_{i-d} is useless. Hence, the maximum number of MACs that can be attached in each packet is d, and instead of just attaching one MAC computed by K_i to each packet, we attach up to d MACs computed using $K_i, K_{i-1}, \cdots, K_{i-d+1}$, respectively. As shown in Fig. 2, the jth packet sent in interval i is

$$\{M_{j,i}, \mathrm{MAC}(\mathrm{K_i}, \mathrm{M_{j,i}}), \mathrm{MAC}(\mathrm{K_{i-1}}, \mathrm{M_{j,i}}), \cdots, \mathrm{MAC}(\mathrm{K_{i-d+1}}, \mathrm{M_{j,i}}), \mathrm{s_{i-d}}\}. \quad (1)$$

Since staggered TESLA uses consecutive and chained key seeds, it inherits the same resilience to packet loss as conventional TESLA.

We now discuss two issues related to the staggered TESLA packet. First, we note that a simple and clever attack, which we shall call the shift attack, may be employed on the above packet format. In the shift attack, the adversary may take advantage of the fact that there is more than one MAC attached to each packet, and make use of the MACs from previous packets and shift them to forge later packets. For example, an adversary can

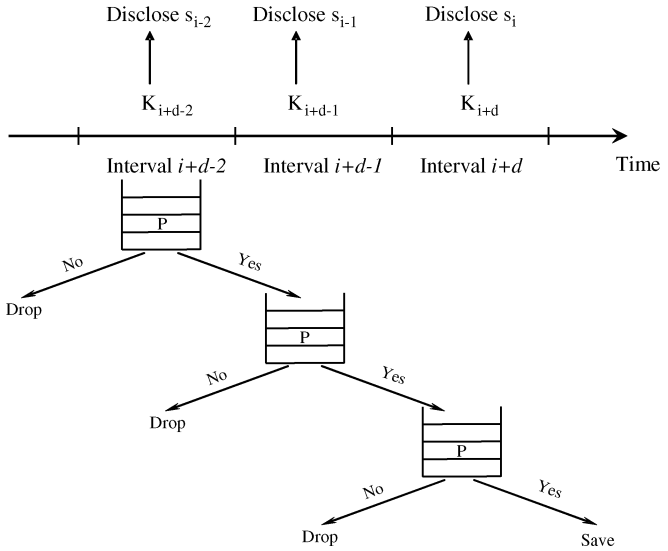


Fig. 3. Events in staggered TESLA and the chained buffer at the receiver. Partially authenticated packets graduate to lower layers of the buffer as the key seeds arrive at the receiver.

store packet j from interval i, as in (1), and use it to forge the packets for interval $i+1$ by sending

$$\{M_{j,i}, \mathrm{MAC}\left(\mathrm{K'_{i+1}}, \mathrm{M_{j,i}}\right), \mathrm{MAC}(\mathrm{K_i}, \mathrm{M_{j,i}}), \cdots, \mathrm{MAC}(\mathrm{K_{i-d+2}}, \mathrm{M_{j,i}}), \mathrm{s_{i-d+1}}\}. \quad (2)$$

All of the MACs will be valid MACs except for the one using the fake K'_{i+1}, which the adversary could not forge. This attack, however, can easily be addressed by incorporating interval numbers and sequence numbers, as is typically done to prevent replay attacks [22] in the implementation when computing the MACs. Consequently, rather than complicate the notation in the remainder of the paper, we stick with the above representation and note that the additional resources needed for appropriate indexing are minimal.

Second, the additional overhead for staggered TESLA is minimal for many typical multicast scenarios. In particular, since MACs are based on symmetric cryptography, they are computationally efficient. Further, MACs produce short-message digests and, therefore, the additional computation and communication requirements introduced by the extra MACs will not cause significant performance degradation. Consider a typical medium-quality video multicast, where the average frame size is 1300 B [23]. In this case, the addition of a few 20-B data fields, corresponding to a SHA-1 MAC, is minimal relative to the actual application data. We further note that one may employ fewer than d MACs, depending on the application's security requirements as well as the bandwidth restrictions of the underlying network to reduce overhead.

B. Multigrade Source Authentication

In staggered TESLA, the receiver-side buffer is a sequence of queues, as conceptually depicted in Fig. 3. When the receiver receives a packet, it puts the packet into the top level of the queue, and graduates the packet to lower layers as additional key seeds arrive and the corresponding MACs are verified. If

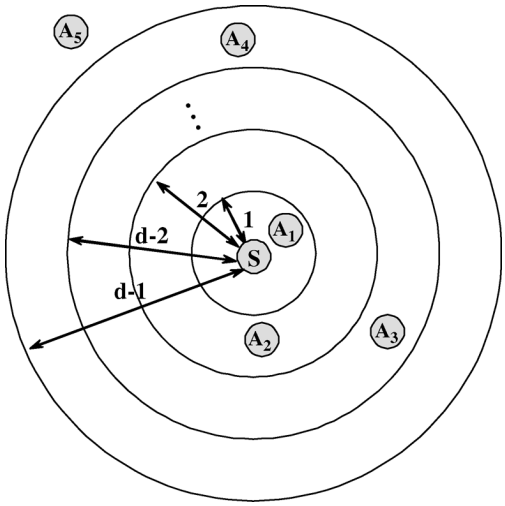


Fig. 4. Adversaries at different locations pose different levels of threats. The receiver can remove false packets from more distant adversaries, such as A_5, sooner than those from closer adversaries, such as A_1. The forge-capable area shrinks as the packets pass authentication at each layer.

any verification fails, the packet is dropped from the queue, while if it passes, the packet becomes more trusted and graduates to the next layer of the buffer. This process repeats until the final key seed involved arrives and complete authentication is achieved. The chained buffer structure is easily implemented by tracking all packets waiting for a key, and updating which key each packet is waiting for following a partial authentication. Hence, the chained buffer we propose does not require any additional overhead compared with the traditional receiver buffer in TESLA.

In staggered TESLA, if an adversary forges a packet for interval i, some of the MACs besides the normal TESLA MAC $\mathrm{MAC}(K_i, M_{j,i})$ are likely to be wrong. Thus, the receiver will most likely be able to discard the forged packet before it would need to check $\mathrm{MAC}(K_i, M_{j,i})$. Further, as a packet successively graduates from a higher layer to a lower layer in the buffer, the likelihood that the packet is trustworthy increases. Thus, the receiver does not have to wait for the seed s_i in order to start authenticating packets. Instead, the receiver can use whatever seeds he or she has received to begin the authentication process and can promptly remove bogus packets. As a result, false packets are removed from the buffer quicker than in conventional TESLA. In contrast, with conventional TESLA, a forged packet will have to remain in the buffer for the complete disclosure delay before its falseness is revealed.

An individual packet that has had only some of its MACs verified is not fully authenticated and, instead, is only partially trustworthy. A packet's trustworthiness is directly related to which MACs have been verified and the amount of MACs employed in staggered TESLA. Further, there is a direct relationship between a packet's position in the buffer and the size of the forge-capable area. Fig. 4 shows the location of the sender and how the forge-capable area changes as key seeds are released. The distance in the figure denotes the relative time delay between the hosts, and for simplicity of discussion, we consider that the network delay characteristics are fixed. Consider a packet that is sent during interval i, which has d MACs appended to it. These MACs are computed using keys $K_{i-d+1}, K_{i-d+2}, \cdots, K_{i-1}, K_i$, respectively. Let us label these MACs as the first, second, $\cdots$, and dth MAC. During time interval $i+1$, the seed s_{i-d+1} is released. The receiver is able to authenticate the first attached MAC after it receives the seed s_{i-d+1}. Since we assume the adversary-receiver link has delay 0, the forge-capable area for the first MAC is the circle of delay radius $d-1$ from the source. Adversary A_5, which is outside the circle, cannot forge packets with a valid first MAC. Thus, if there were an adversary at location A_5, the receiver would be able to remove all bogus packets sent by A_5 from the buffer at this time. However, adversaries within the radius $d-1$ circle (i.e., A_1 to A_4), are able to forge the first MACs for any ith interval packet. Thus, the receiver cannot decide whether those packets are forged packets or not at this time.

At time interval $i+2$, seed s_{i-d+2} is released. Now the receiver can perform authentication on the second MACs and, similarly, the forge-capable area shrinks to a region with radius $d-2$. Now, both adversary A_5 and A_4 are outside the forge-capable area and both of them are unable to forge packets with valid second MACs. The receiver can now remove all packets sent by adversary A_4. Similarly, the forge-capable area shrinks as the packets pass authentication at each layer of the buffer. There is progressively less area from which an adversary could successfully forge packets. Finally, during time interval $i+d$, the seed s_i is released, the forge-capable area has radius 0, and no adversary can forge packets with valid dth MACs. The receiver can fully authenticate the packet.

A packet gains trustworthiness as its forge-capable area shrinks. It is desirable to represent a packet's trustworthiness by a numerical value γ between 0 and 1. Such a quantification for partial authentication can allow for new security policies to be developed whereby partially authenticated packets are accepted if they have a threshold trust level. For example, a multimedia application might have strict QoS delay requirements and a security policy may be specified whereby, if the service quality provided to the user is not acceptable, the application would release additional packets whose γ is above a threshold set by the application designer.

The trust representation γ should be consistent across different staggered TESLA sessions involving different interval sizes and different amounts of MACs. Hence, trust should be defined based on which MACs were used and which MACs were verified. Gambetta [24] defined trust to be the subjective probability that an agent can perform a particular action before that action can be monitored and before it affects a decision. There will be d or fewer MACs employed for each packet in staggered TESLA, and a subset of these MACs will be verified. For staggered TESLA, trust then corresponds to quantifying the likelihood that there are no adversaries that could have forged a particular subset of the MACs.

If we have 3 knowledge of the distribution for the delay τ between the source and a potential adversary, then we could take advantage of such information to define trust. Suppose that the adversarial delay τ has distribution f_τ. Then, for a staggered

TESLA scheme having d disclosure delay, which uses d MACs, and where the first t MACs have been verified, we may define trust as

$$\gamma(t) = 1 - \int_0^{d-t} f_\tau(\tau) d\tau, \quad \text{for } t < d. \tag{3}$$

In the absence of any *a priori* distribution, two natural distributions that we may use are to choose τ to be uniformly distributed over $[0, d]$, or to assume that the network delay corresponds to propagation in two dimensions and places the adversaries uniformly within a circle of radius d. This second choice is suitable for modeling delay in *ad hoc* networks, where a relationship between geographic location and network hop counts has been shown [25]. This leads to a distribution

$$f_\tau(\tau) = \begin{cases} 2\tau/d^2, & \text{for } 0 \le \tau \le d \\ 0, & \text{for } \tau > d. \end{cases} \tag{4}$$

In the case of the first distribution, our trust becomes $\gamma(t) = t/d$. On the other hand, in the second case, the trust becomes $1 - (d-t)^2/d^2$. This definition of trust corresponds naturally our visualization of the forge-capable area as a circular region. In both cases $\gamma(d) = 1$, which corresponds to the trust level associated with full authentication via the conventional TESLA MAC. We note that when measuring trust, we do not need to know the position of the adversary, but only which MACs have been verified.

To further illustrate the relationship between trust, network size, and interval length, let us look at an example. Consider a network with a 400-ms delay between the sender and the receiver, and an 800-ms delay for the key release. If the interval size is 200 ms, then the key disclosure delay is four intervals. There are a total of five levels of trust. However, if the interval size is 100 ms, the key disclosure delay will be eight intervals, and there will be nine levels of trust. There are tradeoffs between the selection of interval size and the number of levels of trust. If the interval size is large, there are fewer intervals and seeds needed. Hence, less communication overhead is needed to transmit those seeds. But at the same time, there are fewer levels of trust. On the other hand, there will be more levels of trust for smaller interval sizes. But this requires a longer key chain and larger communication overhead to distribute key seeds. Applications can select the interval size according to the network condition and security requirements.

The potential damage that can be caused due to the authentication buffer is related to the adversary's location—the closer an adversary is to the source in terms of network proximity, the longer his or her forged packets will remain in the buffer. A coalition of adversaries may attempt a collusion attack [26], [27], where the coalition shares key information with each other in order to facilitate an attack. We note, however, that in a collusion attack, the adversary that is closest to the source is the most important member of the coalition as he or she is the one who will acquire the key seed first. Hence, even if there is a high-speed connection between adversaries, the strength of a collusion attack involving L adversaries is no greater than L times the strength of the closest adversary. Therefore, for the remainder of the paper, we shall only consider the case of a single adversary. The impact of the adversary's location on the receiver's buffer characteristics will be further discussed in Section VI-C. The end result is that staggered TESLA allows the buffer to be more efficiently utilized and provides an advantage against DoS buffer overflow attacks. We will explore this behavior further in Sections V and VI.

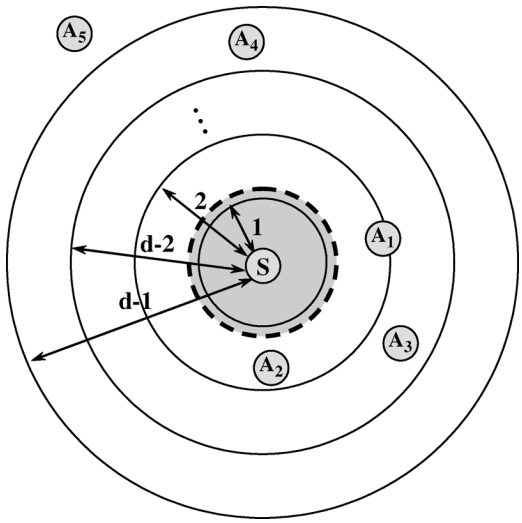


Fig. 5. If there are guarantees that there are no adversaries located within the dashed circle of the source, packets can be fully authenticated one interval earlier in staggered TESLA than in conventional TESLA.

IV. Reduced-Delay Multicast Authentication Schemes

In the previous section, we discussed how partially authenticated packets can be released without waiting for the full authentication delay. We now examine two strategies that reduce the average delay needed for full authentication. The first scheme requires the assumption that the source has a guarantee of the trustworthiness of nearby network entities, while the second approach involves the introduction of additional key distributors, which are synchronized with the source.

A. Staggered TESLA With Proximity Protection

Adversaries at different locations pose different threat levels. The notion of forge-capable areas suggests that complete authentication is possible if we combine partial authentication with complementary forms of information assurance. One possibility, which we refer to as proximity protection, involves a guarantee that adversaries are not located nearby the source in network space. Proximity protection allows us to reduce the full authentication delay since, a few time intervals after the receipt of a staggered TESLA packet, partial authentication will have reduced the forge-capable area to a small enough region that proximity protection will provide the remaining guarantee.

Consider, in Fig. 5, a source with proximity protection around his neighborhood so that it is guaranteed that there are no adversaries within the dashed circle. During interval $i + d - 1$, where key K_{i-1} is released, the forge-capable area shrinks to the region within radius 1, which is included in the dashed circle.

No adversaries can forge the MACs computed using K_{i-1} successfully since they are all outside the forge-capable area of key K_{i-1}. Even though there is still one MAC left to be authenticated for each packet from interval i, the receiver can conclude that all packets that passed the authentic check for key K_{i-1} are actually fully authenticated and release those packets now. In order to save communication overhead, the source even does not need to attach the MAC's corresponding to key K_i for packets from interval i. The larger the area the source can protect, the better we can reduce the full authentication delay. The amount of time intervals that can be reduced for full authentication corresponds to the largest forge-capable area within the protected region.

When specifying a region in network space near the source that can be protected, it is necessary to realize that the network delay will vary from packet to packet in a real network. Consequently, it is important to choose the region that the source can protect based upon the minimal delay that would be experienced by a packet traversing a portion of the network. In particular, since the queuing component of network delay might be 0, the region that can be protected should be decided based upon the nonvariable components composing network delay (i.e., propagation, transmission and minimum processing delay). Another way to look at this is that the protected area must be the intersection of the guaranteed areas for all packets.

In practice, proximity protection can be realized in different ways for different types of networks. The relationship between network delay and hop counts suggests that the source needs to have a guarantee that network entities within a certain hop count are trustworthy. For networks, such as the Internet, this corresponds to having a guarantee that devices within the same access network are trustworthy. For networks, such as wireless ad-hoc networks, hop counts can be related to actual physical distance from the source [25]. Thus, in such networks, network proximity protection becomes equivalent to physically guaranteeing that there are no adversaries within a geographic region around the source. Additionally, proximity protection may be achieved by appropriately employing scheduling and traffic control at intermediate routers (e.g., an overlay network or an ad-hoc network) in order to ensure a specified level of minimum network delay.

Finally, we note that the authentication delay that can be reduced by proximity protection is quantized to multiples of the interval length. If an application needs to further reduce the authentication delay gap between the protected area and the largest forge-capable area, or the guaranteed area is too small to include any forge-capable area, it can use smaller intervals at the expense of additional communication overhead. Further, it might be necessary to shorten the interval length in order to have the resolution to define protected regions on networks where the nonvariable delay component is small.

B. Distributed Key Distributors

We now present a scheme for reducing full authentication that may be used when there are no proximity guarantees. This scheme can be used with traditional TESLA or staggered TESLA. For simplicity, we focus our discussion on applying distributed key distributors to TESLA.

We start by examining the total time that a packet will stay in the buffer in conventional TESLA, then we discuss the factors we can change to reduce delay. Consider a packet sent at time t_i in interval i, which takes l_i time units to arrive at the receiver. The first packet in interval $i+d$ that contains the key seed needed to authenticate the packets from interval i will be sent at time t_{i+d}. It takes l_{i+d} time units for this packet and, hence, the key seed to be delivered. Upon receiving this packet, the receiver can start authenticating packets from interval i. Thus, the total time that the packet from interval i will remain in the buffer is $t_{i+d}+l_{i+d}-t_i-l_i$. Among these four factors, t_i, l_i, and t_{i+d} are unchangeable. We can control l_{i+d}, the time needed for the key to cross the network, by introducing additional key distributors in the network. These key distributors are trusted by the source, and possess a copy of the whole set of key seeds prior to the start of communication. The key distributors must be time-synchronized with the source, and will send out key seeds at the same pace as the source. Synchronization can be accomplished by employing standard methods, such as NTP [28], to synchronize a set of distributed servers. The key distributors do not distribute content, but instead save communication overhead by sending only one key packet for each interval. The source can be thought of as a special key distributor, which sends out keys and data at the same time. The use of the key distributors allows us to partition the network, where each network node belongs to the partition with minimum delay between the key distributor and itself. This reduces the average delay needed to receive the authentication key.

The key distributors can be placed at arbitrary locations in the network, though it is desirable to evenly place the key distributors in the network in order to better reduce the average authentication delay. If the network topology is known and the size of the network is small, the optimal locations can be obtained by exhaustive search. For larger networks, the k-means algorithm [29] can be modified to find the optimal locations for the key distributors. Each object is categorized in one of the k clusters according to the nearest neighbor policy. In our key distributor problem, the positions of the n network nodes yield k centroid points, where we place the k key distributors. However, it should be noted that a slight modification to the k-means algorithm is necessary since we do not have control over the position of the source, and must therefore fix one of the centroid positions, and determine the locations of remaining $k-1$ centroids in order to minimize network delay.

We now outline a modified k-means algorithm for placing the key distributors. As input to the algorithm, we assume that we have knowledge of the relative network delay positioning of each network entity.

1) Begin with an initial choice of $k-1$ nodes, together with the source as the centroid points.
2) Partition the whole set of objects into k clusters using the nearest neighbor policy.
3) Compute the centroid for each cluster and obtained a new set of centroid points, except the one with the source as its centroid.
4) Compute the attribute for the new partition. If it has been changed by a small enough amount since the last iteration, then stop. Otherwise, go to step 2.

Just as in the traditional k-means algorithm, our modified k-means algorithm will converge to a local optima. In reality, network delay is variable, and the positions of the key distributors can be achieved using the estimated average delay.

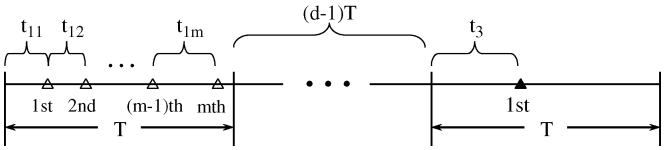


Fig. 6. $d+1$ consecutive intervals at the receiver. We depict the arrivals from the source during the first interval using blank triangles, and denote their interarrival times by t_{1j}. The first packet received in the $d+1$th interval, whether from the source or the adversary, is depicted by the solid black triangle, and arrives after t_3 seconds after the start of the $d+1$th interval.

Finally, we would like to briefly mention an alternative to distributed key distributors. The multiple key distributors are responsible just for transmitting key information. It is possible to replicate full multicast server functionality in the network and have the replicated servers transmit both content and key seeds. This has the effect of cutting the network into smaller networks for both the content distribution and the key distribution functions. Such a strategy is merely running multiple staggered TESLA or TESLA servers and, therefore, we will not consider it further in this paper.

V. Buffer Requirements and Tradeoffs

When using staggered TESLA, choosing an appropriate buffer size becomes an important issue. Too large a buffer size is a waste of resources, while too small a buffer will result in buffer overflow. In this section we revisit staggered TESLA and explore the required buffer size for threat scenarios consisting of different adversarial attack rates. By explicitly calculating the average buffer size needed for the receivers, we provide guidelines for designing the buffer to fit the application and threat environment.

We employ a single adversary with the same network layout as depicted in Fig. 1. Let us consider $d+1$ successive time intervals at the receiver. These intervals correspond to the receipt of packets sent in $d+1$ consecutive intervals, as represented in Fig. 6. We denote the duration of each time interval by T, which is a constant value. Throughout our discussion, we will assume that an adversary forges packets corresponding to the interval that is associated with the latest key seed the receiver knows. We will revisit this assumption in Section VI.

We will break the calculation of the buffer requirements into two parts: first, we will consider packets originating from the source, and then we will consider adversarial packets. After completing these analyses, we combine the two components to get the total average number of packets in the buffer.

For the first part, we assume that the packets sent by the source follow a Poisson process with parameter λ_1,1 and, thus, the interarrival times t_1 are governed by an exponential distribution with parameter $\lambda_1, p(t_1) = \lambda_1 e^{-\lambda_1 t_1}$. We assume that there are a total of m arrivals in the first interval that came from the source, and we denote their interarrival times as $t_{12}, \cdots, t_{1m}$, as depicted by the blank triangles in Fig. 6. Since the interarrivals are exponentially distributed, we may use the memoryless property to define t_{11} to be the time from the start of the first interval to the first arrival, in which case t_{11} has the same distribution as t_1.

During the $d+1$th time interval, the first packet that arrives, which we depict with a solid black triangle, may be either from the source or from the adversary. By the memoryless property of the exponential distribution, the time period from the boundary of the $d+1$th interval to the arrival of the first received packet originating from the source in the $d+1$th interval has the same distribution as t_1. Similarly, if the adversary emits packets as a Poisson process with parameter λ_2, then the time period t_2 from the boundary of the $d+1$th interval to the first received packet originating from the adversary in the $d+1$th interval has exponential distribution with parameter λ_2, $p(t_2) = \lambda_2 e^{-\lambda_2 t_2}$. Hence, the time period t_3 from the boundary of the $d+1$th interval to the first received packet in the $d+1$th interval is the minimum of t_1 and t_2, $t_3 = \min(t_1, t_2)$. Assuming that t_1 and t_2 are independent, then t_3 has exponential distribution with parameter $\lambda_1 + \lambda_2$, i.e. $p(t_3) = (\lambda_1 + \lambda_2) e^{-(\lambda_1+\lambda_2)t_3}$.

Packets originating from the source during interval i can be authenticated when the receiver receives the first packet sent during interval $i+d$ because the packet contains the key seed needed to recover the authentication key K_i. In Fig. 6, all packets received in the first interval will be authenticated after the receiver receives the first packet in the $d+1$th interval. Therefore, the total time W these packets will stay in the buffer are

$$\begin{aligned} 1\text{st} \quad & W = dT + t_3 - t_{11} \\ 2\text{nd} \quad & W = dT + t_3 - (t_{11} + t_{12}) \\ \vdots \quad & \vdots \\ m\text{th} \quad & W = dT + t_3 - \sum_{i=1}^{m} t_{1i}. \end{aligned} \tag{5}$$

The expected value of W is

$$E[W] = \sum_{m=0}^{\infty} E[W|M=m] p_m \tag{6}$$

where p_m is the probability of having m packets from time interval 1. The expected value of W conditioned on the total number of arrivals that originated from the source M is the average of the total time these M packets will stay in the buffer. Thus

$$\begin{aligned} E[W|M=m] = \int \cdots \int & \frac{mdT + mt_3 - \sum_{i=1}^{m}(m-i+1)t_{1i}}{m} \\ & \times p(t_{11}, \cdots, t_{1m}, t_3) dt_{11} \cdots dt_{1m} dt_3. \end{aligned} \tag{7}$$

Since $t_{11}, t_{12}, \cdots, t_{1m}$ are from independent exponential distributions with parameter λ_1, and t_3 has exponential distribution with parameter $\lambda_1 + \lambda_2$, the expected values are $1/\lambda_1$ and $1/(\lambda_1 + \lambda_2)$, respectively. Hence, (7) can be simplified as

$$\begin{aligned} E[W|M=m] &= dT + \frac{1}{\lambda_1 + \lambda_2} - \frac{\sum_{i=1}^{m}(m-i+1)}{m\lambda_1} \\ &= dT + \frac{1}{\lambda_1 + \lambda_2} - \frac{(m+1)}{2\lambda_1}. \end{aligned} \tag{8}$$

Substituting (8) into (6) and noting that M has Poisson distribution with parameter $\lambda_1 T$, yields

$$\begin{aligned} E[W] &= dT + \frac{1}{\lambda_1 + \lambda_2} - \frac{\lambda_1 T + 1}{2\lambda_1} \\ &= dT - \frac{T}{2} + \frac{1}{\lambda_1 + \lambda_2} - \frac{1}{2\lambda_1}. \end{aligned} \tag{9}$$

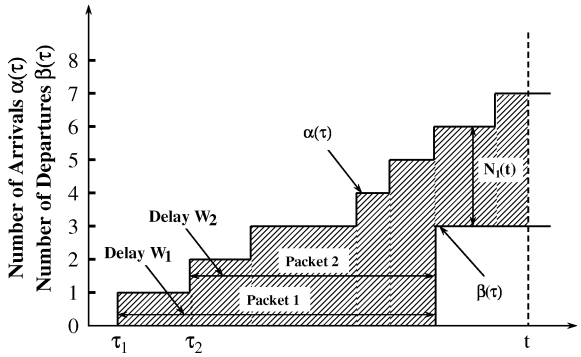


Fig. 7. Arrival and departure process $\alpha(t)$ and $\beta(t)$ for packets arriving at a receiver.

We will let $N_1(t)$ stand for the number of packets that originated from the source and are in the buffer at time t, while we will denote $\alpha(t)$ to be the number of packets originating from the source that the receiver receives and $\beta(t)$ to be the number of packets the receiver authenticates until time t. Further, let W_i be the time spent in the buffer by the ith received packet originating from the source. The time average of N_1 up to time t is

$$N_{1t} = \frac{1}{t}\int_0^t N_1(\tau)d\tau. \tag{10}$$

Usually N_{1t} changes with time t, but it tends to a steady-state value, N_1, as t increases, $N_1 = \lim_{t\to\infty} N_{1t}$. Similarly, the steady-state arrival rate of packets originating from the source during $[0, t]$ is defined as

$$\lambda_1 = \lim_{t\to\infty} \frac{\alpha(t)}{t}. \tag{11}$$

The average time in the buffer spent by a packet that originated from the source is

$$W = \lim_{t\to\infty} \frac{\sum_{i=1}^{\alpha(t)} W_i}{\alpha(t)}. \tag{12}$$

The arrival and removal of the packets sent by the source is shown in Fig. 7. The arrival process follows a single increase model, whereas the removal process is a multiple decrease model since many packets are flushed from the buffer simultaneously. From Little's theorem [30], the relationship between N_1, λ_1, and W is $N_1 = \lambda_1 W$. Assuming ergodicity in the arrival process, we may equate the time average W with the ensemble average $E[W]$ to get $N_1 = \lambda_1 E[W]$.

The derivation of the average time to remove a false packet is similar to the above calculation. The receiver does not need to wait for the full disclosure delay period d to remove false packets in staggered TESLA. Let d' be the number of intervals needed to remove a forged packet. Then, the expected total time W' that a forged packet will stay in the buffer is

$$E[W'] = d'T - \frac{T}{2} + \frac{1}{\lambda_1 + \lambda_2} - \frac{1}{2\lambda_2}. \tag{13}$$

TABLE I
(THEORETICAL) AVERAGE NUMBER OF PACKETS IN BUFFER

Rate(ms)	∞	40	20	10	5	2
#MACs= 1	18.0	35.0	52.5	87.5	157.5	367.5
#MACs= 2	18.0	30.0	42.5	67.5	117.5	267.5
#MACs= 3	18.0	25.0	32.5	47.5	77.5	167.5
#MACs= 4	18.0	20.0	22.5	27.5	37.5	67.5

From Little's theorem, the average number of false packets N_2 can be expressed as $N_2 = \lambda_2 W'$. Again from the assumption of the ergodicity of the arrival process, we can equate W' with $E[W']$. Thus, $N_2 = \lambda_2 E[W']$.

The packets in the buffer either originate from the source or from the adversary. Hence, the total average number of packets N in the buffer is the sum of those originating from the source and those from the adversary

$$N = N_1 + N_2. \tag{14}$$

We calculated the average number of packets in the buffer for different attack rates and for varying amounts of MACs employed in staggered TESLA. The values were calculated according to (14), and are presented in Table I, where the interval length was 200 ms, the delay disclosure was four intervals, and the mean interarrival time from the source was 40 ms. The first line of the table corresponds to the average interarrival time of the adversary's packets in units of milliseconds. An infinite adversarial interarrival time corresponds to no adversary. If we place the adversary at a distant location relative to the source, the single MAC case, which corresponds to conventional TESLA, has $d' = 4$ intervals. Similarly, in this scenario, when we use four MACs, the number of intervals needed to purge forged packets is $d' = 1$. From this table, we see the advantage that staggered TESLA provides as we increase the attack rate.

One way to think of the system is an $M/G/\infty$ queue with two classes of arrivals—one from the source and the other from the adversary. These two arrivals are independent Poisson processes with different parameters and, hence, their sum is simply another Poisson process with parameters equal to the sum of the parameters of the two classes. These two classes of arrivals, however, have different service characteristics. Forged packets have a service time that depends on the availability of authentication keys, while nonforged packets must wait the full disclosure delay to be fully authenticated.

VI. SIMULATIONS AND PERFORMANCE ANALYSIS

We performed a series of event-driven simulations to evaluate the performance of staggered TESLA and techniques to reduce the full authentication delay. The first set of simulations, presented in Sections VI-A–C, studies the multigrade property of staggered TESLA. In these simulations, we assume there is no variability in the link delays. This allows us to deduce the effect of the adversary's network position on the buffering and authentication process. The second set of experiments, presented in Section VI-D, involves a more general network with variable delay links.

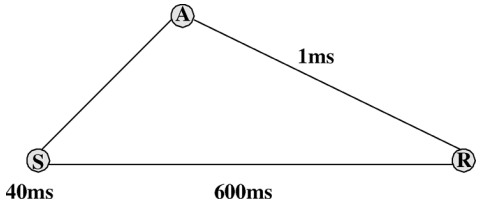


Fig. 8. Arrangement of the source (S), receiver (R), and adversary (A) in the simulations. The connection between each pair of participants represents the aggregate link.

A. Simulations on Multigrade Authentication

The first set of experiments analyzes the improvement to the utilization of the receiver's buffer, as well as the impact that an adversary's location has on the buffer's behavior. We conducted two simulations that involved one sender, one receiver, and one adversary, as shown in Fig. 8. In this setup, we abstracted the possible existence of multiple hops between each entity by representing the connection between each pair by a single, effective link. The first simulation we conducted is designed to show the effective usage of the receiver's buffer and the speed at which the receiver removes forged packets from the buffer when the adversary is at a fixed location. The second simulation shows how the performance of staggered TESLA changes for specific locations of the adversary relative to the source and receiver. In both simulations, we collected statistics for the number of packets in the buffer and calculated the percentage of packets that actually originated from the source. Additionally, we recorded the total time needed to purge a forged packet from the buffer.

For both simulations, we set the length of a time interval to be $T = 200$ ms, and the key disclosure delay to be four intervals. Both the source and the adversary send out packets as a Poisson process. The source sends out packets with an average of interarrival time of 40 ms, which is a typical sending rate for MPEG-4 video [23]. The average interarrival time of packets from the adversary is a parameter in the first simulation, and fixed at 5 ms in the second simulation. The network delay between the source and the receiver was set to 600 ms. We assumed that the adversary has a fast link to the receiver with a delay of only 1 ms. The delay between the source and the adversary is set to 599 ms in the first simulation, and varies in the second simulation.

The objective of a DoS attack is to keep the receiver's buffer as full as possible. We now look at the strategic issues governing the adversary's attack. An adversary has to decide which interval he or she will attempt to forge packets for before he or she transmits those packets. On one hand, an adversary does not want to send "old" packets that have already violated the TESLA security condition as these will be immediately discarded by the receiver. On the other hand, if the adversary knows the key seeds before the receiver, he or she also does not want to release those seeds to the receiver because giving new key seeds to the receiver will help the receiver free the buffer even faster. Third, the adversary also wants to make the bogus packets stay in the buffer as long as possible. Thus, the adversary should forge packets corresponding to the interval associated with the latest key seed that the receiver knows.

In order to reveal the behavior of staggered TESLA under the worst possible threat scenarios, we empower the adversary by giving him or her knowledge of the difference between the sender-to-receiver network delay and the sender-to-adversary-to-receiver network delay. From the knowledge of the network delays, the adversary can figure out the newest key the receiver will know when he or she receives forged packets. The adversary should then transmit packets from the interval that corresponds to the release of that key. If the adversary has some of the keys to calculate the attached MACs, those MACs will pass the authentication check. If the adversary does not have the keys, he or she will fake those MACs with random bits. Those MACs will fail in the authentication check. The closer the adversary is to the source, the sooner he or she receives the key seeds and, thus, can attach more valid MACs to the packets, requiring a longer period for the receiver to remove the forged packets. In the worst case, if the adversary knows all of the key seeds except the latest, it will take the receiver the full disclosure delay to flush bogus packets from the buffer.

B. Performance Analysis of Staggered TESLA

We now examine different sending rates for the adversary and the effect these rates have on the performance of staggered TESLA. In order to gauge the efficiency of the staggered MACs to remove packets from the buffer, we set the buffer size to be sufficiently large so that buffer overflow does not occur for all adversarial transmission rates. We measured the number of packets in the receiver's buffer and calculated the proportion of packets in the buffer that originated from the source. Additionally, we computed the total time needed to remove a false packet from the buffer. The simulation was run for 50 s, long enough for the system to achieve steady-state. We compared the performance for different amounts of MACs in staggered TESLA. Since the key disclosure delay was four intervals in the simulation, the maximum number of MACs that could be employed in each packet was 4. Note that when only one MAC is attached to each packet, the situation is precisely conventional TESLA.

In the simulation, the adversary is set to be relatively far away from the source (at a fixed delay of 599 ms). In order to demonstrate the behavior of staggered TESLA during a normal traffic scenario and to illustrate the potential damage that our adversary can cause, the first 5 s involved only the source, and the adversary commences his or her DoS attack after that.

Fig. 9 shows a realization of the number of packets in the buffer for the first 30 s, when the average interarrival time of packets from the adversary is 5 ms. Before the start of the adversary's DoS, the number of packets in the buffer is about 18. The number of packets in the buffer sharply increases as the adversary starts sending forged packets for all cases. The dashed lines depict the average number of packets in the buffer before the start of the adversary's DoS, while the solid lines depict the average during the DoS. When the receiver receives a packet which does not contain a new key seed, the packet will be put into the buffer. When a packet is received that provides the key seed for a new key, all of the packets in the buffer with MACs claiming to have been created using the new key will undergo authentication verification. If the adversary does not have the new key when he or she forges packets, those packets are proven

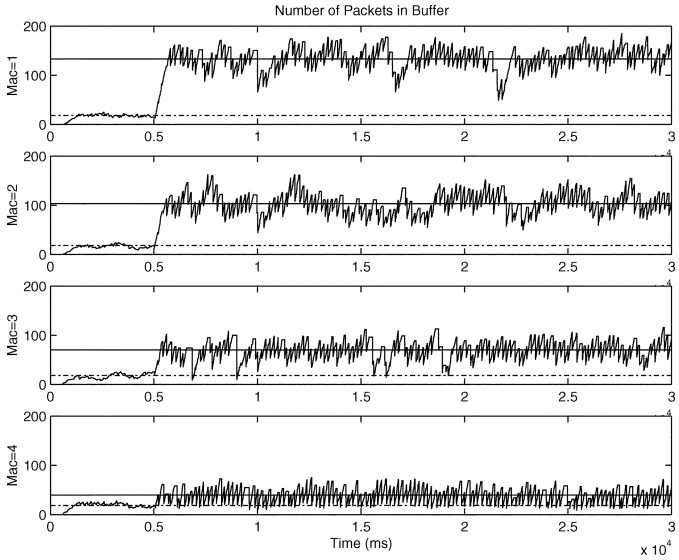


Fig. 9. Number of packets in the buffer in the first simulation, where the source–adversary link has a delay of 599 ms. The four curves correspond to different amounts of MACs employed in staggered TESLA. The adversary starts the DoS attack at 5 s.

TABLE II
AVERAGE NUMBER OF PACKETS IN BUFFER

Rate(ms)	∞	40	20	10	5	2
#MACs= 1	18.1	33.5	48.6	75.0	133.1	307.1
#MACs= 2	18.1	28.3	40.7	58.8	101.9	227.1
#MACs= 3	18.1	24.8	30.5	44.1	69.8	149.2
#MACs= 4	18.1	20.1	24.3	29.4	38.9	68.8

to be false in this layer's authentication check and are then discarded. If the new key is the last key used to compute the MACs in a packet from the source, the packet will be completely authenticated.

It is clear from Fig. 9 that when multiple MACs are employed in staggered TESLA, the number of packets in the buffer are much lower during a DoS than in the case of conventional TESLA. A full-fledged version of staggered TESLA (in this case, employing all four MACs) is able to significantly reduce the average amount of packets in the buffer, even compared to the cases when only two or three MACs are used. In Table II, we present the averaged values of the number of packets during the DoS for a time period of 45 s. The first line of Table II is the average interarrival time of packets sent by the adversary, in units of milliseconds. ∞ corresponds to no adversary, and can be identified with the first 5 s of the simulation. Columns further to the right represent more powerful adversaries that are capable of conducting their DoS attack at higher attack rates. In all cases, the number of packets in the buffer increases as the adversary's sending rate increases.

The true advantage of staggered TESLA is revealed when we examine the results within each column. For a fixed column (i.e., when the sending rate is fixed), the number of packets in the buffer is lower when there are more MACs in each packet. A more enlightening phenomenon is observed when we increase

TABLE III
AVERAGE PERCENTAGE OF TRUSTED PACKETS IN BUFFER

Rate(ms)	∞	40	20	10	5	2
#MACs= 1	1	0.56	0.40	0.24	0.14	0.06
#MACs= 2	1	0.64	0.47	0.32	0.19	0.08
#MACs= 3	1	0.74	0.60	0.43	0.28	0.13
#MACs= 4	1	0.88	0.80	0.68	0.55	0.34

TABLE IV
AVERAGE TIME TO PURGE A FORGED PACKET

Rate(ms)	∞	40	20	10	5	2
#MACs= 1	N/A	724.5	723.5	726.4	724.1	725.5
#MACs= 2	N/A	522.8	521.2	524.0	523.8	522.7
#MACs= 3	N/A	327.5	325.8	327.4	326.9	323.7
#MACs= 4	N/A	125.1	124.7	124.3	121.9	125.8

the attack rate of the adversary. For example, examining the columns for an attack rate of 40 and 2 ms, which corresponds to an increase in the attack rate by a factor of 20, we see that the number of packets in the buffer increases roughly by a factor of 10 for conventional TESLA, but only by a factor of 3 for staggered TESLA with four MACs employed. Comparing the four cases in the table, full-fledged staggered TESLA has the best performance.

Staggered TESLA not only decreases the number of packets in the buffer compared to conventional TESLA, but also improves buffer utilization efficiency. In Table III, we calculated the average percentage of packets in the buffer that originated from the source. In all cases, the utilization efficiency drops as the adversary's sending rate increases. For a fixed sending rate, the efficiency increases as we use more MACs and shows the improvement that the full-fledged staggered TESLA provides compared to TESLA. Further, for full-fledged staggered TESLA, the buffer utilization drops slower as we increase the transmission rate than it does for conventional TESLA. Overall, these results mean that staggered TESLA will provide improved resilience to buffer overflow attacks.

Overall, the use of multiple, staggered MACs in delayed key disclosure decreases the buffer requirements and more efficiently uses the buffer. These improvements are due to the fact that the receiver removes false packets faster in staggered TESLA than in conventional TESLA. This can be explicitly seen in Table IV, where we present the average time needed to remove a false packet from the buffer. When there is only one MAC attached to each packet, it takes the receiver the full-delay disclosure time to remove false packets. Since the key disclosure delay is four, it takes the receiver four intervals to remove a false packet. Because some packets arrive earlier in an interval and some arrive later, the average time to remove a false packet is around 720 ms. When there are two MACs in each packet, it takes the receiver three intervals to remove a false packet and the average time to flush false packets is around 520 ms. The decrease in time is due to the fact that the adversary does not have both K_i and K_{i-1} when forging

TABLE V
AVERAGE NUMBER OF PACKETS IN BUFFER

Delay(ms)	∞	600	500	400	200	1
#MACs= 1	18.1	133.1	156.4	157.5	158.0	157.2
#MACs= 2	18.1	101.9	115.6	116.8	118.7	148.4
#MAC= 3	18.1	69.8	78.6	77.9	109.3	146.9
#MACs= 4	18.1	38.9	53.2	71.4	110.8	150.6

packets. When the number of MACs increases to three, the number of intervals needed to remove a false packet decreases to two. Finally, when there are four MACs appended to each packet, it only takes the receiver one interval to remove forged packets, which yields an average buffer time of around 120 ms. In this case, when the adversary forges the packets, he or she does not know any of the keys used to compute the MACs (i.e., K_i, K_{i-1}, K_{i-2}, and K_{i-3}).

C. Impact of the Locations of Adversaries

We conclude from the above discussion that the source should attach d MACs in each packet to optimize the performance when the adversary is "relatively" far away. We now examine the relationship that the adversary's position has upon staggered TESLA. We emphasize that the advantages provided by staggered TESLA do not depend on the ability of either the source or the receiver to locate the adversary's relative position, nor does it depend on the ability to formally map out a forge-capable region in the network. Rather, the performance advantages follow strictly from the use of multiple MACs.

The second simulation was conducted to analyze the effect of the adversary's position. The adversary's DoS behavior was fixed throughout all simulations as a Poisson source with an average sending rate of a 5-ms delay between consecutive packets. The adversary was placed at different, constant network delay distances from the source. We measured the number of packets in the buffer for different locations for the adversary. The average number of packets in the buffer is shown in Table V. The first row in the table is the source-to-adversary-to-receiver delay in units of milliseconds. It was assumed that the adversary had a fast connection with which to attack the receiver and, thus, the adversary-receiver delay was fixed to 1 ms. Thus, the adversary becomes progressively closer to the source as we move from left to right on the table. The ∞ delay corresponds to no adversaries.

From this table, we see that for scenarios where the adversary is further away from the source, there is improved buffer behavior as we use more MACs. On the other hand, when the adversary is closer to the source, there is little advantage to employing multiple MACs as the number of packets for different amounts of MACs is practically the same. A second observation that can be made from this table is that the number of packets in the buffer for different MACs can be divided into an amount of clusters that are roughly $1 + \mathrm{Delay}/T$. For example, in the case where $\mathrm{Delay} = 1$ ms, there is a single cluster centered at 150 packets, while for $\mathrm{Delay} = 400$ ms, there appears to be three clusters: one at 157.5, one at 116.8, and one centered around 75. Similar observations can be made when one examines the

TABLE VI
AVERAGE TIME TO PURGE A FORGED PACKET

Delay(ms)	∞	600	500	400	200	1
#MACs= 1	N/A	724.1	707.0	704.8	705.0	704.8
#MACs= 2	N/A	523.8	507.2	504.3	504.8	664.4
#MACs= 3	N/A	326.9	307.2	303.9	461.8	663.0
#MACs= 4	N/A	121.9	177.0	262.8	464.4	666.7

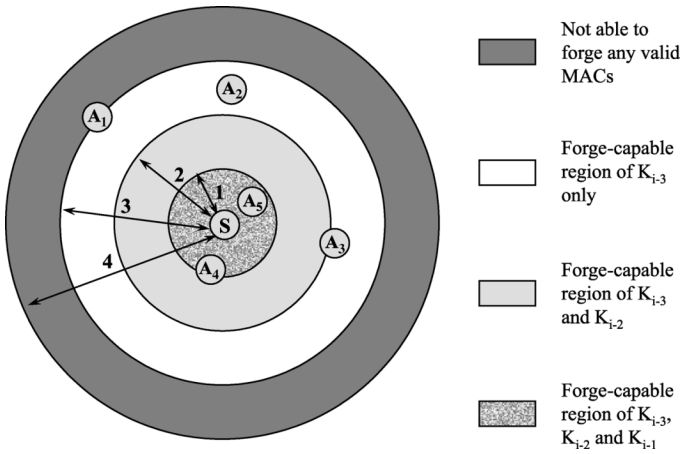


Fig. 10. Position of different classes of adversaries and their corresponding forge-capable areas for staggered TESLA. Here, the key disclosure delay is $d = 4$.

average time needed to purge a forged packet, as presented in Table VI.

As we discussed earlier, keys sent out at different times will result in different forge-capable areas. The position of the adversary determines how many valid MACs he or she can forge when he or she sends out the forged packets. Let us consider a staggered TESLA packet, sent during interval i, that consists of four MACs, such as

$$P_i = \{M_{j,i}, \mathrm{MAC}(\mathrm{K_i}, \mathrm{M_{j,i}}), \mathrm{MAC}(\mathrm{K_{i-1}}, \mathrm{M_{j,i}}), \cdots, \mathrm{MAC}(\mathrm{K_{i-3}}, \mathrm{M_{j,i}}), \mathrm{s_{i-4}}\}.$$

As shown in Fig. 10, when the path from the source to the receiver via adversary A_1 is 600 ms (three intervals), A_1 is just outside the forge-capable area of K_{i-3}. A_1 does not have any of the keys needed to compute the MACs when he or she forges the packets. Thus, when there are four MACs attached in the packets, the receiver can remove those forged packets from A_1 when he or she receives K_{i-3}. It takes the receiver only one interval to remove false packets. We have depicted the sequence of events leading to the removal of forged packets in this scenario in Fig. 11(a). During time interval $i+3$, the adversary gets P_i and has K_{i-4}. At the same time, the receiver has received P_i and has K_{i-4}. Recall that we assumed a powerful adversary who knows the state of the receiver he or she is attacking, and that the adversary will therefore forge packets corresponding to the interval associated with the latest key the receiver knows. Thus, during interval $i+3$, the adversary will create forged packets P_i'. The adversary did not know any of the keys K_i, K_{i-1}, K_{i-2}, or K_{i-3} and, hence, P_i' will only stay in the receiver's buffer for

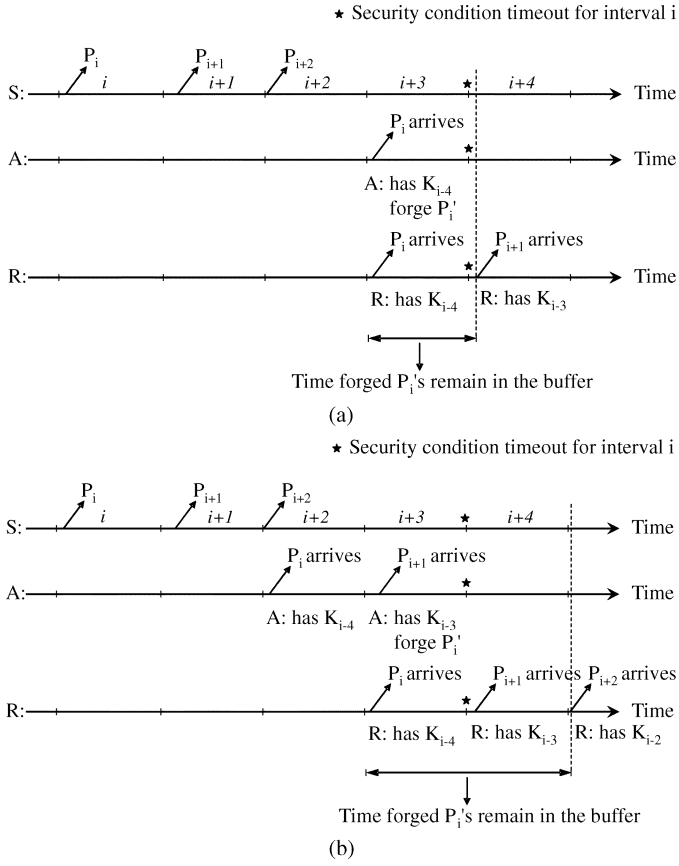


Fig. 11. Sequence of events in staggered TESLA. (a) The case when the source (S)-adversary(A)-receiver(R) delay is 600 ms. (b) The case when the S-A-R delay is 400 ms.

one interval. In contrast, if only three MACs were used in the packets, the receiver will only be able to remove false packets when he or she receives the seed for K_{i-2}, and the receiver must wait for two intervals before removing false packets. In the extreme case, for conventional TESLA, it will take the receiver four intervals to remove false packets.

Consider adversary A_3, whose source-adversary-receiver delay is two intervals. He or she is outside the forge-capable area of K_{i-2}, but inside the forge-capable area of K_{i-3}. If the source attaches four MACs, then A_3 is able to forge the fourth MAC correctly. We present the sequence of events for A_3 in Fig. 11(b). A_3 receives P_i during interval $i+2$, but will create a forged packet P_i' during interval $i+3$. Because A_3 knows K_{i-3} during time interval $i+3$, the receiver will not be able to reject P_i' until interval $i+5$, when it gets the seed to calculate K_{i-2}. Thus, it is actually $\mathrm{MAC}(K_{i-2}, M_{j,i})$ that provides the ability to remove forged packets and, hence, $\mathrm{MAC}(K_{i-3}, M_{j,i})$ does not help to remove packets any faster. This explains why attaching three and four MACs gives roughly the same performance for an S-A-R delay of 400 ms in Table V.

Other locations for the adversary follow the same patterns. For adversary A_4, which is one interval away, attaching two, three, or four MACs gives roughly the same result. Finally, for adversary A_5, who can get the key seeds as soon as the source releases them, only one MAC is able to authenticate packets and, thus, all cases give approximately the same result as TESLA.

An interesting observation can be made if we examine adversary A_2, who is 2.5 intervals from the source. A_2 is inside the forge-capable area of key K_{i-3}. At first glance, the case should be similar to the scenario for A_3 and have only three levels. However, since the adversary receives s_{i-3} in the middle of an interval, all packets forged before the adversary receives the seed will have wrong fourth MACs. For some adversarial packets, all four MACs are useful for removing packets, while for others, only three MACs are useful. Thus, there are four levels for the number of packets in the 500-ms column in Table V, though the improvement of attaching four MACs over three MACs is not as large as for the 600-ms delay case.

The theoretical values in Table I are close to the simulation results in Tables II and V. There are two factors affecting the differences. First, the theoretical calculations assumed the key seed is always available at the beginning of an interval for the adversary. In reality, this is not the case, and this effect is more pronounced when the adversary is far away. The adversary might receive P_i shortly into the interval, but until that time, any forged P_i' will use an incorrect s_{i-d} and, thus, will be immediately dropped by the receiver since it will fail the key seed verification. This results in the simulation having slightly lower values, as seen when comparing the 600-ms cases with the values from Table I. Second, the theory calculations assumed that during each interval, the key seed is always available in the first packet that arrives at the receiver, though in actuality, the first few packets might fail seed verification. Consequently, packets from interval $i-d$ remain in the buffer for a slightly longer period of time, and this effect can be seen in the case where four MACs are used. These two effects appear in different locations in the tables. Overall, the discrepancies are small, which suggests that the theoretical calculations can serve as a good guideline for determining buffer requirements.

D. Simulation on Reducing Authentication Delay

The second set of experiments compares the full authentication delay and communication overhead of our two schemes described in Section IV. The simulations were conducted in the ns2 simulation environment with the network configuration shown in Fig. 12. The network is comprised of 64 stationary nodes located on an 8 × 8 grid. The distance between adjacent nodes is 50 m, and the communication range and sensing range are set to be 50 and 100 m, respectively. Thus, nodes can only communicate with their adjacent neighbors. Node 56 is set to be the source, which sends out packets as a Poisson source with 40-ms average delay between successive packets. For simplicity, broadcast is used as the traffic dissemination pattern in the simulations. Each node only forwards newly received packets and discards all old ones. There is a fixed 25-ms processing delay at each node before forwarding each packet. In addition to the variable queuing delay provided by ns-2, we added a random delay that was uniformly distributed between 0 and 10 ms to reduce collisions. The payload of each packet is 1300 B, corresponding to a typical medium-quality video [23]. We chose 802.11b for the *ad hoc* network and, thus, the link bandwidth is 11 Mb/s. The farthest node (i.e., node 7) is 14 hops away via the shortest path and has a shortest path network delay of around

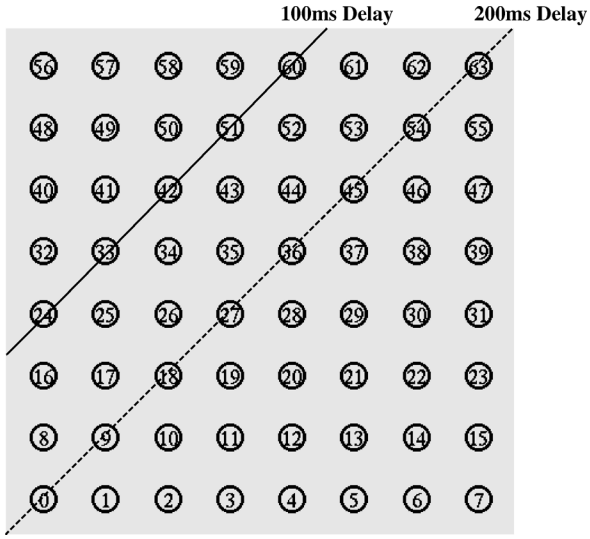


Fig. 12. Grid network topology of 64 nodes. If the source can protect the region left to the solid line, then the average authentication delay can be reduced by 100 ms. The dashed line is the assurance boundary of 200 ms.

400 ms. The addition of queueing delay and use of an alternative path may result in longer delay. The interval size is 100 ms, and key disclosure delay is 800 ms.

In Fig. 12, the five-hop neighbors of the source will have a delay greater than 100 ms. If the source can employ proximity protection for the region to the left of the solid line (which denotes the 100-ms delay line), then it is not necessary for the receivers to check the authenticity of the last MAC for full authentication. Further, it is not necessary to attach the last MAC in order to save communication overhead.

The positions of the key distributors are determined by the minimum sum of network delay according to the partition made by the key distributors. We placed the distributors by conducting a search to find the best locations for up to four key distributors. For the case of only one key distributor, either node 20 or 29 will be the solution. When two key distributors are placed, four combinations give the same result—nodes 10 and 38, nodes 11 and 45, nodes 11 and 46, and nodes 18 and 38. Only one choice of locations is available for three key distributors, namely, nodes 14, 18, and 45. Either nodes 10, 22, 34, and 53 or nodes 13, 17, 43, and 46 can be chosen for the four key distributors scenario. The source sends out packets as a Poisson process with 40-ms average interrarival time, while the key distributors send out one key packet per interval.

The simulation results are shown in Table VII, where we compare TESLA, full-fledged staggered TESLA, and staggered TESLA with a proximity protection of 100- and 200-ms delay, and the use of distributed key distributors with up to four key distributors placed in their optimal positions. The simulation was run for 50 s of network time while, in all cases, steady-state was achieved in only a few seconds of simulation time. The five columns stand for the packet size in bytes, average authentication delay in milliseconds, maximum authentication delay in milliseconds, packet delivery ratio, and the bandwidth consumed in bytes at each node for each data packet compared to TESLA. Each data packet consists of 24-B physical-layer convergence protocol (PLCP) header and preamble, 24-B MAC header, 4-B frame control sequence (FCS), 20-B IP header, 16-B released key, MACs (each being 20 B), and the payload. For the proximity protected delay of 100 ms, only seven MACs are attached to each packet, while only six MACs are attached to each packet for the proximity protection delay of 200 ms. For key packets used by the key distributors, there is no payload and no MACs are attached, producing a key-bearing packet of size 88 B.

TABLE VII
COMPARISON OF REDUCED-DELAY AUTHENTICATION SCHEMES

	Size	Avg	Max	Deliver	Bandwidth
TESLA	1408	780.33	1021.99	95.82%	100%
Staggered	1548	783.06	1050.44	95.39%	109.4%
100ms	1528	676.22	939.86	95.09%	107.7%
200ms	1508	580.64	842.08	95.35%	106.6%
Distributed1	1408	663.52	1009.58	95.19%	101.1%
Distributed2	1408	635.96	976.39	95.28%	101.3%
Distributed3	1408	616.61	954.54	95.61%	101.9%
Distributed4	1408	607.52	923.17	95.52%	102.0%

As presented in Table VII, both the proximity protection and distributed key distributor schemes can significantly reduce average authentication delay. The maximum authentication delay can also be reduced in most cases. Proximity protection for 100/200 ms can reduce the average authentication delay by about 100/200 ms. With only one key distributor added in the network, the average authentication delay will decrease by about 120 ms, which is slightly better than proximity protection for 100 ms. Adding extra key distributors will further decrease the average authentication delay. Due to collisions, in all cases, the packet delivery ratio is about 95%. Compared to TESLA, there is an additional 9%, 7%, and 6% communication overhead compared with full-fledged staggered TESLA, staggered TESLA with a proximity protection of 100 ms, and staggered TESLA with proximity protection of 200 ms. By comparison, there is only a 1%–2% additional communication overhead for key distributors. It should be noted, however, that the reduced communication cost for distributed key distributors does not capture the overhead needed to maintain synchronized key distributors, or the cost needed to install the key distribution functionality at different locations in the network.

VII. CONCLUSION

In this paper, we introduced the notion of multigrade authentication and presented an approach by which multiple degrees of trust can be incorporated into multicast authentication schemes based on delayed key disclosure. We developed a multigrade multicast authentication scheme, known as staggered TESLA, that employed multiple, staggered authentication keys that are used in creating the MACs for authenticating a packet. As a result, the receiver may partially authenticate a packet by using those authentication keys it has prior to the arrival of new key seeds. The use of these staggered MACs not only provides varying levels of authentication, but also reduces

the delay needed to filter forged packets, thereby resulting in more efficient utilization of buffer resources compared to conventional TESLA. Theoretical results were derived that provide design guidelines for determining the appropriate buffer size. Further, theoretical and simulation results showed that the use of multiple MACs and, hence, multiple grades of authentication, allow the receiver to flush forged packets quicker than conventional TESLA. As a result, staggered TESLA provides an advantage against a DoS attack as it requires an adversary to attempt a DoS at a higher attack data rate than is necessary in conventional TESLA. With the complementary forms of information assurance, staggered TESLA can further reduce full authentication delay. We also examined a second strategy for reducing full authentication delay by introducing additional key distributors in the network. Simulations showed that staggered TESLA with proximity protection, as well as the use of additional key distributors, is able to reduce authentication delay compared to TESLA, with a minor increase in communication requirements.

ACKNOWLEDGMENT

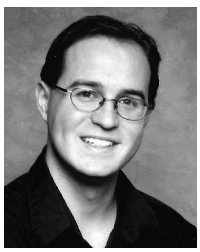
The authors would like to thank A. Perrig for valuable discussions regarding TESLA and staggered TESLA, particularly some discussion about the shift attack on staggered TESLA.

REFERENCES

[1] S. Paul, *Multicasting on the Internet and Its Applications*. Norwell, MA: Kluwer, 1998.
[2] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC, 1997.
[3] R. Gennaro and P. Rohatgi, "How to sign digital streams," in *Proc. Advances in Cryptology: Crypto*, 1997.
[4] C. K. Wong and S. Lam, "Digital signatures for flows and multicasts," *IEEE/ACM Trans. Netw.*, vol. 7, no. 4, pp. 502–513, Aug. 1999.
[5] P. Rohatgi, "A compact and fast hybrid signature scheme for multicast packet authentication," in *Proc. 6th ACM Conf. Computer Communications Security*, 1999, pp. 93–100.
[6] J. M. Park, E. K. P. Chong, and H. J. Siegel, "Efficient multicast packet authentication using signature amortization," in *Proc. IEEE Symp. Security Privacy*, 2002, pp. 227–240.
[7] J. M. Park, E. K. P. Chong, and H. J. Siegel, "Efficient multicast stream authentication using erasure codes," *ACM Trans. Information System Security*, vol. 6, no. 2, pp. 258–285, 2003.
[8] C. Karlof, N. Sastry, Y. Li, A. Perrig, and D. Tygar, "Distillation codes and applications to dos resistant multicast authentication," in *Proc. Network Distributed System Security Symp.*, 2004, pp. 37–56.
[9] A. Lysyanskaya, R. Tamassia, and N. Triandopoulos, "Multicast authentication in fully adversarial networks," in *Proc. IEEE Symp. Security Privacy*, 2004, pp. 241–255.
[10] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: a taxonomy and some efficient constructions," in *Proc. INFOCOMM*, 1999, pp. 708–716.
[11] A. Perrig, "The BiBa one-time signature and broadcast authentication protocol," in *Proc. 8th ACM Conf. Computer Communication Security*, Nov. 2001, pp. 28–37.
[12] S. Xu and R. Sandhu, "Authenticated multicast immune to denial-of-service attack," in *ACM Symp. Applied Computing*, 2002, pp. 196–200.
[13] S. Cheung, "An efficient message authentication scheme for link state routing," in *Proc. 13th Annu. Computer Security Applications Conf.*, Dec. 1997, pp. 90–98.
[14] R. Anderson, F. Bergadano, B. Crispo, J. Lee, C. Manifavas, and R. Needham, "A new family of authentication protocols," *ACMOSR: ACM Operating Syst. Rev.*, vol. 32, no. 4, pp. 9–20, 1998.
[15] F. Bergadano, D. Cavagnino, and B. Crispo, "Chained stream authentication," in *Proc. 7th Annu. Workshop on Selected Areas in Cryptography*, Aug. 2000, pp. 144–157.
[16] F. Bergadano, D. Cavagnino, and B. Crispo, "Individual single source authentication on the mbone," in *Proc. IEEE Int. Conf. Mutlimedia Expo.*, Aug. 2000, pp. 541–544.
[17] B. Briscoe, FLAMeS: Fast, Loss-Tolerant Authentication of Multicast Streams, http://www.labs.bt.com., Tech. Rep., BT Research, 2000.
[18] A. Perrig, R. Canetti, D. Song, J. D. Tygar, and B. Briscoe, "TESLA: multicast source authentication transform introduction," IETF Working Draft, draft-ietf-msec-tesla-intro-01.txt.
[19] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in *Proc. Network Distributed System Security Symp.*, Feb. 2001, pp. 35–46.
[20] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "The TESLA broadcast authentication protocol," *RSA Cryptobytes*, vol. 5, no. 2, pp. 2–13, 2002.
[21] A. Perrig, R. Canetti, J. D. Tygar, and D. Song, "Efficient authentication and signing of multicast streams over lossy channels," in *Proc. IEEE Symp. Security Privacy*, 2000, pp. 56–73.
[22] C. Kaufman, R. Perlman, and M. Speciner, *Network Security: Private Communication in a Public World*. Upper Saddle River, NJ: Prentice-Hall, 1995.
[23] F. H. P. Fitzek and M. Reisslein, "MPEG-4 and H.263 video traces for network performance evaluation," *IEEE Netw.*, vol. 15, no. 6, pp. 40–54, Nov./Dec. 2001.
[24] D. Gambetta, *Trust: Making and Breaking Cooperative Relations*. Oxford, U.K.: Basil Blackwell, 1988.
[25] D. Niculescu and B. Nath, "Dv-based positioning in ad hoc networks," *Telecommun. Syst.*, pp. 267–280, 2003.
[26] J. Fan, P. Judge, and M. H. , "Hysor: group key management with collusion-scalability tradeoffs using a hybrid structuring of receivers," in *Proc. 11th Int. Conf. Computer Communications Networks*, 2002, pp. 196–201.
[27] H. Chu, L. Qiao, K. Nahrstedt, and H. Wang, "A secure multicast protocol with copyright protection," *ACM Comput. Commun. Rev. J.*, vol. 32, pp. 42–60, 2002.
[28] D. Mills, "Network Time Protocol (Version 3) Specification, Implementation and Analysis," IETF RFC [Online]. Available: http://www.ietf.org/rfc/rfc1305.txt
[29] A. Gersho and R. M. Gray, *Vertor Quantization and Signal Compression*. Norwell, MA: Kluwer, 1992.
[30] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1992.

Qing Li (S'05) received the B.S. degree in physics from Zhongshan University, Guangzhou, China, in 1996, and the M.S. degree in electrical and computer engineering in 2002 from Rutgers University, Piscataway, NJ, where she is currently pursuing the Ph.D. degree at the Wireless Information Network Laboratory (WINLAB) and the Electrical and Computer Engineering Department.

Her research interests include network security, multicast authentication, and computer networking.



Wade Trappe (S'98–M'02) received the B.A. degree in mathematics from The University of Texas at Austin in 1994 and the Ph.D. degree in applied mathematics and scientific computing from the University of Maryland, College Park, in 2002.

Currently, he is an Assistant Professor in the Wireless Information Network Laboratory (WINLAB), Rutgers University, Piscataway, NJ. His research interests include wireless security, wireless networking, multimedia security, and network security. He is coauthor of the textbook *Introduction to Cryptography with Coding Theory* (Prentice-Hall, 2001).

Dr. Trappe received the George Harhalakis Outstanding Systems Engineering Graduate Student Award from the University of Maryland. He is the recipient of the 2005 Best Paper Award from the IEEE Signal Processing Society. He is a member of the Association of Computing Machinery (ACM).