Round-Efficient Broadcast Authentication Protocols for Fixed Topology Classes

Haowen Chan, Adrian Perrig
Carnegie Mellon University
Pittsburgh, Pennsylvania, USA
haowenchan@cmu.edu, perrig@cmu.edu

***Abstract*—We consider resource-constrained broadcast authentication for n receivers in a static, known network topology. There are only two known broadcast authentication protocols that do not use asymmetric cryptography, one-time signatures, multi-receiver MACs, or time synchronization [1], [2]. Both these protocols require three passes of a message front traversing the network. We investigate whether this amount of interaction can be improved efficiently for specific common topology classes, namely, linear topologies, tree topologies and fully connected topologies. We show modifications to the protocols allowing them to complete in just two passes in the linear and fully connected cases with a small constant factor increase in per-node communication overhead, and a further optimization that achieves the equivalent of just a single pass in the linear case with $O(\log n)$ increase in per-node communication overhead. We also prove new lower bounds for round complexity, or the maximum number of consecutive interactions in a protocol. We show that protocols with efficient per-node communication overhead (polylogarithmic in n) must require at least $2 \log n$ rounds in any topology; this implies that our two-pass protocol in the fully-connected topology requires the fewest possible passes, and this bound is asymptotically tight for the full-duplex communication model. Furthermore, we show that communication-efficient protocols must take asymptotically more than $2 \log n$ rounds on trees; this implies that that there are some tree topologies for which two passes do not suffice and the existing three-pass algorithms may be optimal.**

***Keywords*-Multicast Authentication, Broadcast Authentication, Linear Topology, Path Topology, Fully Connected Topology**

I. Introduction

Broadcast authentication is a critical security primitive in computer networks. Due to the high computational overhead of asymmetric cryptography and the high communication costs of one-time signatures and multi-receiver message authentication codes (MMACs), *signature-free broadcast authentication protocols* have received a great amount of attention in the research literature. Instead of a single cryptographic construction transmitted from sender to receivers, such protocols rely on multiple interactions among the sender and the set of receivers. Of these, TESLA and related protocols assume secure time synchronization and involve long message buffering times. There are two remaining types of efficient broadcast authentication protocols: *hash-chain protocols* based on the "Guy Fawkes" protocol first proposed by Anderson et al. [1] and a *hash-tree protocol* that we described in previous work [2]. Both of these are three-pass protocols, requiring message exchanges that traverse the network a total of three times. Typically, the payload message and an initial authenticator is disseminated from the sender to the rest of the network; then a return pass of messages is initiated from the edges of the network, flowing back towards the sender in a "converge-cast" traffic pattern; finally a third wave of messages flows outwards from the sender to the rest of the network. The fact that the only known broadcast protocols that do not require time synchronization all require three passes across the network motivates the question of whether three passes is universally minimal for such protocols. We show that this is not the case by describing communication-efficient two-pass versions of both algorithms for certain general classes of network topologies. At this point it may seem like there is no further potential for improvement: a true single-pass protocol implies the ability of each receiver to verify the broadcast message upon reception; this seems unlikely without the use of costly signatures or MMACs. However, we show that this intuition is misleading. For the linear topology (where all receivers are arranged in a simple path) we can achieve a number of rounds of interaction *functionally equivalent* to only taking a single pass across the network: the optimization allows every receiver to authenticate the message within the time it takes to reach the farthest receiver from the sender. In addition to these optimizations, we also show new theoretical lower bounds proving that our optimizations do indeed achieve the lowest possible number of passes in the topologies that we consider, as well as showing that some classes of trees do require more than two passes, and thus the three-pass algorithms may indeed be optimal in general scenarios. This implies that analyses which consider separate optimizations for separate network topology classes are a necessary methodology for optimizing protocol structure.

The concept of reducing the number of "passes" across the network is intuitive; we quantify this more precisely. Fundamentally, we are interested in optimizations for *reducing dependencies* and *increasing parallelism* in terms of interactions, such that the protocol requires a smaller number of consecutive communication events. The metric of choice is thus *round complexity*: we model the protocol as taking a number of rounds, where, in each round, every node exchanges information with at most one other node (this is the standard "one-port" model used in the theory of distributed systems [3], in which each machine has a single network interface). Round complexity is a favored measure of the time complexity of a protocol in the analysis of distributed algorithms [4] because it is a fundamental property of the protocol, reflecting how the protocol structures its data/message dependencies: a protocol with low round complexity necessarily has a shallow dependency graph. As an intrinsic property of the protocol, round complexity can thus be expressed solely in terms of n, the number of nodes in the network, and, unlike more specific measures of delay, remains constant regardless of the size of the broadcast data payload and the particular physical or link-layer characteristics of the network.

To measure the communication overhead of our protocols, we use worst-case per-node communication, or *congestion*, as a metric (this is more formally defined in Section III). Low communication congestion yields lower per-node energy consumption, which is important for many wireless network applications. For completeness, we also consider *total* communication overhead.

Our exploration of the design space for these protocols thus yields various trade-offs between round complexity and communication overhead. Consequently, different protocols and optimizations will be most efficient in terms of more direct performance measures (such as latency, energy cost, or usage of the medium) depending on the different deployment environments.

We examine three topology classes covering most easily-characterizable topology types in practice: paths, trees and complete graphs. The path or *linear* topology occurs frequently in wireless networks both directly in such examples as networks deployed along corridors and roadways and indirectly as a multi-hop routing path between a sender and receiver. Trees are of prime importance in broadcast, since spanning tree construction can occur implicitly in any broadcast pattern: the sender is at the root of the tree, while the parent of each receiver d_i is the node that first forwarded the broadcast message to d_i. The ease and generality of defining such a spanning tree explains why many broadcast protocols are defined only on trees [2], [5]. Trees can also occur directly in sensor networks which are often deployed in a hierarchical structure with a base station at the root [6]. The complete graph or *fully connected* topology occurs frequently in Internet-scale distributed systems such as grid computing and peer-to-peer networks, where any node can communicate (via the Internet) with any other node.

Contributions: While there has been much work on reducing the number of rounds of interaction in general cryptographic protocols (such as secure multiparty computation), to the best of our knowledge, we present the first analysis on reducing the number of rounds of interaction in broadcast authentication protocols deployed over network topologies. We show the feasibility of this avenue of research by deriving a variety of new optimizations and bounds for round complexity in specific topology classes with n receivers. For the linear topology, we show an algorithm that completes in the equivalent of a single pass (n rounds), three times less than the round complexity of any currently known protocol. This optimization causes $O(\log n)$ communication congestion overhead. We show another algorithm that completes in $2n$ rounds with constant congestion overhead; and a general parameterization for 2-pass protocols that yields at most $n + \frac{n}{(2^k-1)}$ round complexity for an additive k-factor congestion overhead increase. We prove that any protocol (in any topology) that achieves polylogarithmic congestion overhead must require at least $2 \log n$ rounds. For the fully-connected topology under our communication model, we show that this lower bound is achievable with an algorithm that completes in $2\lceil \log n \rceil + 1$ rounds with $O(\log n)$ congestion overhead. We also show a round complexity lower bound of $2.44 \log n - 2.33$ rounds for polylogarithmic congestion authenticated broadcast in trees. These are the first proven bounds linking the fundamental trade-offs between round complexity and per-node communication in broadcast authentication protocols.

II. Related Work

In prior work [2], we describe using hash trees over MACs for broadcast authentication. The construction is only described for a tree topology; precise round complexity and congestion optimizations for specific topologies are not investigated. The general Guy Fawkes protocol framework is proposed by Anderson et al. [1]; Bergadano et al. propose CSA, a variant using explicit synchronization with acknowledgments [7]. CSA uses per-receiver acknowledgments; Yao et al. improve this for tree-based networks by proposing the construction of an authenticated aggregate acknowledgment using hashes [5]. Heer et al. suggest using a variant of Guy Fawkes for authenticating two-party messages while simultaneously securing message forwarding on the forwarding path [8]. In this protocol, message authentication is only achieved between the two endpoints; the forwarding nodes only achieve resistance against forwarding extraneous amounts of spurious traffic and cannot authenticate the integrity of the actual message content being exchanged.

Perrig et al. propose TESLA [9], which replaces explicit synchronization with loose time synchronization. There have been a variety of improvements to TESLA, all requiring some form of time synchronization [10], [11]. Zhu et al. propose a lightweight variant for one-hop communications where messages are considered authentic if immediately followed by a valid hash chain value [12]; this approach does not work for multi-hop broadcasts. Luk et al. propose families of broadcast authentication mechanisms [13], but the communication overhead of their one-time signature schemes can be quite substantial.

Multicast MACs (MMACs) are another major class of broadcast authentication mechanisms. An inexhaustive list of work on MMACs follows. Desmedt et al. propose a polynomial-based multicast authenticator [14], which is generalized by Safavi-Naini and Wang [15]. Canetti et al. propose an MMAC by concatenating bits of multiple MACs [16]. Zhang et al. suggest adding perturbations to improve the security of multi-MAC constructions [17]. All MMACs involve communication overhead proportional to the number of node compromises that the scheme tolerates. Boneh et al. show that this overhead is a lower bound for all schemes which do not involve digital signature properties [18]. Stream signatures involve amortizing the cost of a signature over a large number of packets [19], [20], [21], [22], [23]. These approaches either require the sender to know the data stream beforehand or use expensive one-time signatures, or require the receiver to buffer a large number of packets before it can reconstruct the authenticator, and are hence more limited in their applicability than general broadcast authentication techniques. Zhu et al. implement a variant of the multiple-MAC approach in their interleaved hop-by-hop packet filtering system, essentially requiring t individual MACs to protect against up to t malicious nodes [24].

III. Problem Definition

We consider a network consisting of a sender node s and n receiver/destination nodes $d_1, \ldots, d_n$. The network topology is represented by an undirected graph $G = (V, E)$ where $V = \{s, d_1, \ldots, d_n\}$ and an edge connects any two nodes that can communicate (all communication links are bidirectional).

We consider communication in the *single-port* model, where each node has only one network interface and thus can only communicate with only a single neighbor at a time. We also do not consider local broadcast modes of communication (where a single node broadcasts the same message to all its neighbors) because the protocols we consider are sensitive to packet loss; this implies the strong desirability of a link-layer acknowledgment and retransmission mechanism, which essentially reduces a local broadcast to the equivalent of a series of point-to-point communications (in terms of acknowledgments). Rather than model link-layer acknowledgments as separate communication events, for simplicity we consider only bidirectional point-to-point communications in this model. Specifically, each node can only communicate with one neighbor at any given point in time, but this point-to-point communication can involve both sending and receiving.

We assume that each receiver knows the number of receivers n and its own index in the enumeration of the receivers (i.e., node d_1 knows that its index is 1, node d_2 knows that its index is 2, and so on). Also, since we consider topology-specific optimizations, the implicit assumption is that the senders and receivers also know the general class of the topology (e.g., linear, tree, or fully connected).

The sender s shares a unique secret key K_i with each receiver node d_i. The nodes will not use asymmetric-key cryptography in the protocol and rely on symmetric-key methods for authenticity and integrity.

In the broadcast authentication problem, the sender s wishes to send a series of messages $M_1, M_2, \ldots, M_m$ to all receivers such that each receiver can check that each message is authentic, i.e., M_i was truly sent by s. Some unknown fraction of the receivers may be malicious and may behave in arbitrary ways to subvert the protocol. The goal of the adversary is to cause some legitimate receiver to accept some forged M_i' that was never sent by the sender s. We do not consider denial-of-service attacks, where a legitimate message M_i that is sent by s is rejected by a legitimate node due to the malicious actions of the adversary. We assume that the messages contain sequence numbers such that message-replay attacks are infeasible; specifically, each M_i contains a unique sequence number identifying it as the ith message.

We consider several evaluation metrics. The first metric is *communication congestion cost*. Let $c(v)$ be the total amount of communications (transmission or reception) in the entire protocol performed by some node v. Then the congestion of the protocol is $\max_{v \in V} c(v)$, i.e., the greatest amount of communications performed by any single node. We also consider the *total communication cost* incurred by the protocol over all nodes in the network. Since we only wish to consider protocol overhead, we disregard the size of the message payload (i.e., the number of bits in each M_i). Communication overhead is measured assuming all hash values, single-block ciphertexts, message authentication codes and pseudorandom function ranges are of identical length (e.g., 128 bits). Since these are the only values exchanged in the protocols, we measure communication overhead in terms of the number of *cryptographic values* exchanged; this maps linearly to the actual number of bits (e.g., exchanging 4 hash values involves an exchange of $4 \cdot 128 = 512$ bits).

The second metric is the *round complexity* of the protocol. In each round, every node is allowed to exchange information (both ways) from only one other node. The round complexity of the protocol is defined as the total number of such rounds of interaction needed to authenticate a single broadcast message, starting from when the sender s initiates the protocol in the first round up until the latest round by which time all receiver nodes have authenticated the message. Since we are concerned only with correct authentication and not denial of service, we measure only *honest* round complexity. That is, for the purposes of this metric, we do not consider malicious actions specifically aimed at increasing the number of rounds (such as refusing to respond, or connecting to an incorrect communication partner, etc). Conceptually, it can be easier to think of all nodes performing the protocol in lock-step, with all nodes switching connection partners simultaneously at each round: for clarity, this is how we will describe our protocols. In practice, no synchronization is necessary for these protocols: nodes can attempt to establish connections as soon as they are able to do so in the protocol; if their partner is not ready (e.g., it is busy with an existing connection, or waiting on a dependency), then this connection attempt can fail or block until the partner node is ready to connect.

Round complexity is related to actual protocol latency when the amount of data exchanged per round remains relatively small, such that the cost of transferring the data does not strongly dominate over the cost of establishing the link. To take this into account, we also consider the *maximum per-round data transfer*, as measured by the amount of *data* exchanged in the largest single-round message exchange in the protocol. To remain practically viable, a round-complexity optimization should not increase this quantity by a large factor.

Even though some of the motivation of our topology cases are drawn from wireless applications, we do not explicitly consider the effect of wireless medium access contention on the latency of the protocols. There are two reasons for this. Firstly, we focus primarily on round complexity, which is an *intrinsic* property of a broadcast authentication protocol which always holds regardless of the characteristics of the application on which it is deployed. Medium access contention and wireless interference, on the other hand, are extrinsic properties that vary depending on the specific physical deployment characteristics of the application and do not hold as generally. Secondly, in terms of specific effects, we note that under the linear topology, if we model interference as occurring within a radius r of each pair of communicating nodes, this only increases the round complexity and communication congestion by at most an *additive* factor of $O(r)$ under our optimizations (see Section V-A2). This reinforces the observation that medium interference does not fundamentally change the general results derived in this paper.

IV. Background

We consider two classes of algorithms for broadcast authentication for arbitrary topologies: the hash tree based schemes and the hash chain based schemes. We first provide a general description of these algorithms, and then show optimizations and bounds for the various topology classes that we consider in the subsequent sections.

A. Hash Tree Broadcast Authentication

To authenticate a message M to all nodes, the sender could send a different $\text{MAC}_{K_i}(M)$ to each receiver d_i. This has poor communication congestion due to the large number of MACs involved. Instead, we can compute a hash tree over these authenticator values: since only the sender knows all

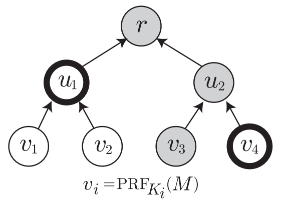

Figure 1. Hash Tree of PRF values. Authentication path of v_3 is highlighted (u_1, v_4).

keys needed to compute the leaves of this hash tree, the adversary cannot derive the root value. Note that the idea of using a hash tree to batch many *MACs* for a single message is completely different from the common idea of using a hash tree to batch many *messages* together to save authentication costs (e.g., Wong and Lam's scheme [23]).

In prior work, we described the hash tree based broadcast authentication protocol for tree topologies [2]. The sender constructs a *binary hash tree* T over a set of n leaf values which contain the authentication codes for the message M. The leaf values are $\{\text{PRF}_{K_i}(M) | i = 1, \ldots, n\}$ where PRF denotes a keyed pseudorandom function accepting arbitrary length inputs (e.g.,Jutla's construction [25]). The hash tree T is constructed over this set of leaf values by repeatedly generating parent vertices to unify adjacent subtrees of the same height. For two subtrees with root vertices c_1, c_2, respectively, a parent vertex p is generated using the rule $p = H(c_1 \| c_2)$ where H is a collision-resistant hash function. This process is repeated until all the vertices are in a single tree. Figure 1 shows such a hash tree; each arrow indicates a hash dependency of a parent vertex on its (two) children.

Given the root vertex r of a hash tree, we can verify the inclusion of a given leaf value u by recomputing all hash tree vertices on the path from u to r, or the *authentication path* of u. The authentication path of vertex v_3 is highlighted in Figure 1. Each receiver d_i knows the position of its respective $\text{PRF}_{K_i}(M)$ in the hash tree, and thus knows the structure of the authentication path from $\text{PRF}_{K_i}(M)$ to r.

Security follows from the observation that the root vertex r and the authentication path together can act as an authenticator for M to d_i. This is because an adversary cannot derive an acceptable root vertex value r' for an unknown leaf vertex value u: a good candidate for r' must work for many possible values of u; finding such a value implies the ability to generate hash collisions for H (a formal proof of this security property is available [26]).

In the original protocol for tree topologies, the authentication paths are reconstructed in two-passes. The leaves of the network perform a convergecast towards the sender allowing internal network nodes to compute internal vertices in the hash tree; subsequently, the sender initiates a wave of messages spreading out towards the leaves resulting in the

Algorithm 1 Hash Tree Broadcast Authentication

```
Sender $s$:
  construct hash tree $T$ over leaf values $\mathrm{PRF}_{K_i}(M)$
  $r$ := root vertex of $T$.
  disseminate $(M, r)$ to all receivers
Each receiver $d_i$:
  check that $M$ is fresh (e.g., inspect seq. num), otherwise abort.
  release $\mathrm{PRF}_{K_i}(M)$ to the network.
  collaborate to reconstruct valid authentication path $P_i$
  check $P_i$. If valid, accept $M$.
```

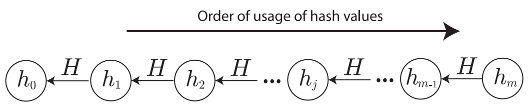

Figure 2. Hash chain

dissemination of all authentication paths. In Sections V-A and VI-B, we show that we can speed this process up considerably for the specific topology classes that we consider in this paper.

B. Hash Chain Broadcast Authentication

Hash chains are used for broadcast authentication in such protocols as the Guy Fawkes schemes (an example of which is the Chained Stream Authentication algorithm or CSA) [7] and in TESLA [9]. The basic structure of these algorithms are similar and can be sketched as follows.

On initialization, the sender generates a hash chain of some predetermined length k by randomly selecting a secret seed value h_k and then iterating a pre-image resistant hash function H to generate a chain of k values such that $h_{i-1} = H(h_i)$ for $i = k-1, k-2, \ldots, 0$. Figure 2 shows a hash chain. Receivers are loaded with the final hash chain value h_0 during initialization.

To authenticate the ith message m_i, the sender broadcasts $(m_i, \mathrm{MAC}_{h_i}(m_i))$. Once all receivers have received $\mathrm{MAC}_{h_i}(m_i)$, the sender broadcasts h_i. Receivers can check that $H(h_i) = h_{i-1}$ and also that the MAC is correct given m_i and h_i. If so, they accept the message.

The security of the scheme follows from the synchronization requirement that h_i is released only after all receivers have stopped accepting candidates for $\mathrm{MAC}_{h_i}(m_i)$. The protocol thus temporally separates MAC generation from MAC verification; at some point in the protocol, it is ensured that h_i becomes useless for MAC generation, and thus h_i can be safely released to facilitate MAC verification. The different variants of hash chain based schemes use different techniques to ensure this temporal separation; TESLA uses loose time synchronization which implies additional assumptions and protocol overheads, while CSA/Guy Fawkes uses explicit acknowledgments which are expensive in terms of communication overhead. In Section V-B we describe several optimizations to speed up and to reduce the communication overhead of explicit acknowledgments.

C. Hash Tree vs Hash Chain: Trade-offs

The hash-chain schemes typically have lower congestion and memory overhead than hash-tree schemes since only one MAC value needs to be stored as opposed to a whole authentication path. On the other hand the hash tree scheme has the advantage of being able to easily recover from missed reception since the PRF values of the hash tree scheme can be selectively revealed by the sender without compromising the security of the protocol; in hash-chain schemes, if any receiver did not receive or respond to all protocol messages, the protocol must be restarted from scratch. A related advantage of hash-tree schemes is the ability to perform authenticated multicast to any subset of the receivers; hash-chain schemes cannot exclude a subset of receivers since when the hash chain key is released, it can potentially be used to forge messages to receivers in the excluded subset.

V. Linear Topology

We now examine optimizations of the hash tree and hash chain protocols for the linear network topology. G is a single path starting at s and ending at d_n, in the sequence $(s, d_1, d_2, \ldots, d_n)$. In this topology, since the nth receiver is n hops away from the sender, any protocol for disseminating M must take at least n rounds.

There are numerous applications of broadcast authentication in a linear topology. For example, wireless networks deployed in a linear environment naturally assume a linear topology. Examples of such networks include sensor and wireless networks deployed along corridors, roadways, tunnels and pipelines. In such environments, all applications for broadcast authentication benefit from our optimizations. Specific examples of such applications include network-key updates in Zigbee, notifications to allow joining/leaving of devices, and network-wide data query messages.

General sensor networks are often logically arranged in a tree with the base station at the root [6], [27]. Paths naturally occur in tree topologies as the subgraph of all the nodes between the root and any leaf. Hence our algorithms are useful for securing broadcast communications to all nodes along any root-leaf path. A specific example of such communications include node join/leave notifications. In the event of a new node joining a tree topology as a leaf, all its ancestors up to the root may need to be authentically notified of the change in membership of their respective subtrees; the notification would take the form of a broadcast along a linear topology from the base station to all affected ancestors. A similar argument applies for nodes leaving the topology and reorganizations of the tree due to sporadic mobility.

More generally, the linear topology also occurs as a *routing path* between any two endpoints. Broadcast authentication on the routing path can then be used for functions related to route maintenance and data transport. For example, endpoints may wish to inform forwarders that they are still actively using this routing path, or to request a certain traffic rate on the route. Endpoints may also send out authenticated packet loss statistics to help forwarders determine if they are responsible for packet dropping bottlenecks. In scenarios where a group key is shared between all nodes on the path (e.g., for the purposes of preventing injection of packets from nodes outside of the forwarding path), authenticated broadcast may be used to maintain consistency in sequence numbers or, in high traffic situations, to update the path key periodically to prevent key overuse.

A. Hash Tree Based Schemes

In the basic hash tree based scheme [2], the protocol proceeds in three passes. The first pass consists of a broadcast message front expanding outwards from the sender; this disseminates the message M and the hash tree root r to all nodes. The second pass is a convergecast message front starting at the leaves of the network topology and converging towards the sender; this allows internal nodes in the topology to compute internal hash tree vertices. The third front is another outgoing message front from the sender and facilitates the dissemination of the computed internal hash tree vertices to the rest of the network for the reconstruction of each node's respective authentication paths.

A naive implementation of the basic protocol on a linear topology will result in a round complexity of $3n-2$ as the first pass takes n hops to traverse the network while the remaining two passes are from the nth receiver back to the first and vice-versa. The maximum amount of data transferred in a round is $\lceil \log n \rceil$ hash values (the length of an authentication path in a hash tree). The communication congestion of the basic protocol is $\lceil \log n \rceil + 2$ hash values (+1 due to minor asymmetries in the tree, and +1 for the root r).

We show an optimization that reduces the round complexity to $2n-1$ with no change in communication overheads and only a small constant increase in maximum per-round data transfer, and a further optimization that can achieve optimal n round complexity by doubling both the congestion overhead and the maximum per-round data transfer.

1) Optimization for $2n-1$ round complexity: The first optimization is based on the following observation: as soon as (M, r) is received by a node i, the value of $\text{PRF}_{K_i}(M)$ can immediately be released to the (untrusted) network to facilitate the derivation of authentication paths for other nodes. In other words, each node can individually start to reconstruct the hash tree as soon as it has received the payload message M and does not have to wait for other nodes to also receive M. This allows us to "piggy-back" the dissemination of (M, r) onto the two passes that reconstruct the hash tree, thus reducing round complexity to $2n-1$.

The optimization is as follows. For a linear topology of n receivers $d_1, \ldots, d_n$ with node d_1 being closest to the sender, the hash tree is constructed with leaf vertices, from left to right, $(v_1, \ldots, v_n)$, with $v_i = \text{PRF}_{K_i}(M)$ where K_i is the key shared between the sender and d_i. We define the *left off-path vertices* L_i for a hash tree leaf v_i as the set of all sibling vertices that are to the left of the path from i to the root vertex r; similarly the *right off-path vertices* R_i of v_i is the set of all sibling vertices that are to the right of the path. For example, in Figure 3, the left off-path vertices of vertex 4 are vertices 3 and 9; similarly, there is one right off-path vertex, 14. Given v_i, L_i and R_i, the authentication path from v_i to the root can be computed.

We show that L_{i+1} can be easily computed from L_i and v_i. Intuitively, we can simply insert v_i to L_i and repeatedly compute hash tree vertices as far up the authentication path of v_i as possible. This yields the deepest left-child in the authentication path of v_i. For example, in Figure 4, given L_4 (vertices 3,9) and vertex 4, we can first compute vertex 10 and then vertex 13, which is the first left-child in the authentication path $(4, 10, 13, 15)$. Vertex 13 is exactly the left off-path vertex of vertex 5. More generally, let h be the height of the hash tree. If L_i has no vertex at maximum depth h, then we can just add v_i to L_i and obtain L_{i+1}. Otherwise, let k be the deepest level of the hash tree which has no vertices in L_i. Then L_i has vertices at all depths from $k+1$ to h. Let them be $u_{k+1}, u_{k+2}, \ldots, u_h$ respectively. Now L_{i+1} has a vertex u_k at level k: this vertex is root of the largest subtree that contains v_i as the right-most child, i.e., it is the deepest left-child in the authentication path of v_i. The path from v_i up to u_k has each of the $u_{k+1}, u_{k+2}, \ldots, u_h$ as left-children; hence we can compute u_k:

$$u_k = H(u_{k+1} \| H(u_{k+2} \| \cdots H(u_{h-1} \| H(u_h \| v_i)) \cdots))$$

All vertices of L_{i+1} at depths less than k are the same as those at the same depth in L_i. A symmetrical reasoning applies to computing R_{i-1} from R_i and v_i. The algorithm thus completes in two passes: in the outgoing pass, nodes compute left off-path vertices for their downstream neighbors; in the incoming pass, nodes compute right off-path vertices for their upstream neighbors. The message and authenticator (M, r) are piggybacked onto the outgoing pass. The algorithm is shown in Algorithm 2; an example is illustrated in Figure 3.

The communication congestion of this protocol remains $\lceil \log n \rceil + 2$ hash tree vertices. The round complexity of the protocol is exactly $2n-1$ rounds since the protocol proceeds in two passes of the linear topology and the second pass does not need to reach the sender. Total communications remains unchanged ($n\lceil \log n \rceil$ hash values) since this algorithm is essentially a re-ordering of the messages in the original

Algorithm 2 2-pass Hash Tree Protocol

```
Sender $s$ sends $(M, r)$ to $d_1$
for $i$ = 1 to $n$ do
    Node $d_i$ checks freshness of $M$ (e.g. inspect seq. num), otherwise abort.
    Node $d_i$ computes $L_{i+1}$ from $L_i$ and $v_i = \mathrm{PRF}_{K_i}(M)$
    Node $d_i$ sends $(M, r), L_{i+1}$ downstream to Node $d_{i+1}$
end for
for $i$ = $n$ down to 1 do
    Node $d_i$ computes $R_{i-1}$ from $R_i$ and $v_i = \mathrm{PRF}_{K_i}(M)$
    Node $d_i$ sends $R_{i-1}$ upstream to Node $d_{i-1}$
end for
```

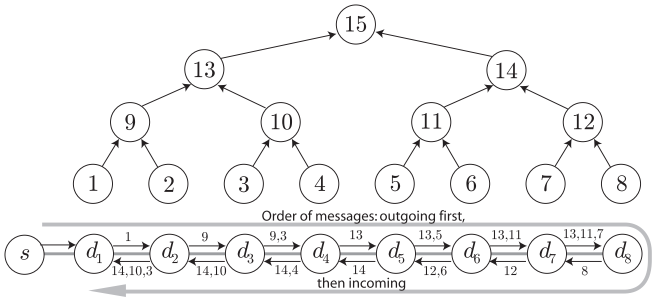

Figure 3. Example of Algorithm 2 with 8 nodes. Numbers indicate transmitted hash tree vertices.

protocol. The maximum per-round data transfer increases by one to $\lceil \log n \rceil + 1$ due to the piggy-backing of r on the outgoing pass.

2) Optimization for n round complexity: The optimization of Section V-A1 reduced the number of passes of the broadcast authentication protocol from three passes down to two. We now present an additional optimization which reduces the round complexity of the algorithm to the equivalent of a single pass, i.e., n rounds. This optimization makes the protocol optimally fast in terms of round complexity, since all broadcast protocols on the linear topology require at least n rounds.

The optimization is parameterizable to trade off round complexity and congestion, and is applicable to all 2-pass authentication protocols. The intuition is that we can perform a variant of divide-and-conquer on the receiver set. We observe that a 2-pass protocol takes $2n - 1$ rounds due to the round-trip time from the sender to the farthest receiver and back. To reduce the size of this round-trip, we can partition the set of n receivers into two contiguous subsets R_1 and R_2. The first subset R_1 contains the first m receivers $\{d_1, \ldots, d_m\}$ (for some selected value of m) and R_2 contains the rest of the receivers $\{d_{m+1}, \ldots, d_n\}$. We then run the protocol separately for each subset. The sender computes different authenticators T_1, T_2 for each receiver subset R_1, R_2. Recall that for the hash-tree scheme, the authenticators are the roots of the hash trees com-

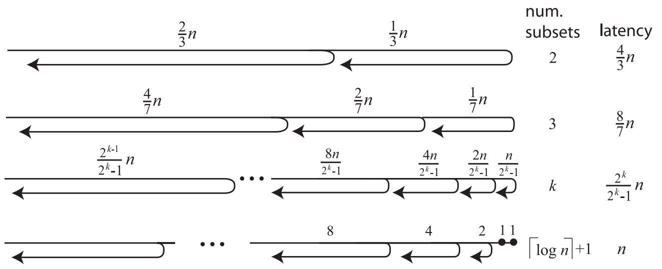

Figure 4. Partitioning the receiver set

puted over the PRF values of the receiver set, so in this case T_1 is the root of the hash tree computed over $\mathrm{PRF}_{K_1}(M), \ldots \mathrm{PRF}_{K_m}(M)$, while T_2 is the root of the hash tree computed over $\mathrm{PRF}_{K_{m+1}}(M), \ldots \mathrm{PRF}_{K_n}(M)$. Receiver subset R_1 performs the protocol using T_1, and forwards T_2 to R_2 as part of its own outgoing pass; once the last node d_m receives T_2, it is forwarded to d_{m+1} in the next round to allow R_2 to perform the protocol. Receiver subset R_1 will take $2m$ rounds: $2(m-1)$ rounds to pass messages within the receiver set and an extra round to receive the first message from s and another for r_m to forward T_2 to R_2. Receiver subset R_2 will take no more than $2(n - m) + m = 2n - m$ rounds since it has $n - m$ receivers and needs m rounds before it receives T_2 forwarded across the m nodes of set R_1. Setting $m = \frac{2}{3}n$ causes both subsets to complete at the same time ($\frac{4}{3}n$ rounds) and minimizes the worst case round complexity.

Figure 4 shows how partitioning the receiver set yields lower overall round complexity. The more subsets we use, the lower the round complexity, at the cost of greater congestion due to the larger number of authenticators that have to be forwarded to the respective receiver groups. For k subsets $R_1, R_2, \ldots, R_k$, worst case round complexity is minimized by having each successive receiver subset be half the size of the previous subset, i.e., $|R_{i+1}| = \frac{1}{2}|R_i|$ for $i = 1, \ldots, n-1$, with the smallest subset having $n/(2^k - 1)$ receivers. This results in an overall round complexity of $n + n/(2^k - 1) - 1$ rounds. For values of n which are not divisible by $2^k - 1$, we need to round down the subset boundaries to the next largest integer value. This may introduce up to one extra round in each subset. Hence the round complexity is bounded by $n + n/(2^k - 1)$ rounds.

Each receiver subset R_i must receive a separate authenticator T_i to execute its respective authentication protocol; these authenticators must be forwarded through the first group R_1, adding to its congestion. For k groups, we thus have a worst-case of $\lceil \log n \rceil + k + 1$ hash values of communication congestion at R_1. Due to the additional overhead of carrying the extra k authenticators across up to n hops each, the total communication overhead increases by up to nk cryptographic values, thus an upper bound on total communications is $n(\lceil \log n \rceil + k)$ hash values.

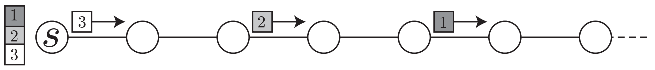

Figure 5. 3 groups of values forwarded separately for an extra 4 rounds

The maximum per-round data transfer also increases by k cryptographic values to $\lceil \log n \rceil + k + 1$.

Since the optimization achieves $n + n/(2^k - 1)$ round complexity for k subsets, setting $k = \log n$ suffices to achieve a round complexity of at most $n + 1$. To achieve optimal n round complexity, we can use a slightly different way of partitioning the receiver subsets as follows. Let $|R_k| = |R_{k-1}| = 1$, and each preceding subset double in size, e.g., $|R_{k-2}| = 2, |R_{k-3}| = 4$, and so on until R_1 contains the remaining receivers. For example, a receiver set of size $n = 35$ can be divided into 7 subsets of size $3, 16, 8, 4, 2, 1, 1$ respectively. It can be seen that, except possibly for R_1 (which may complete early if n is not a power of 2), all subsets complete their respective protocols at the same time: after n rounds. This method of partitioning yields $k = \lceil \log n \rceil + 1$ subsets and involves a total of $2\lceil \log n \rceil + 2$ congestion in the hash tree protocol. Total congestion is $2n(\lceil \log n \rceil + 1)$ hash values.

Application of this family of optimizations increases the maximum per-round data by $\lceil \log n \rceil + 1$ hash values. This expression grows only slowly with n (e.g., 176 bytes for a network with 1000 nodes). Furthermore, none of the values contributing to this data transfer are mutually dependent in terms of processing or forwarding. Hence, in the few applications for which per-round overhead is a factor, these values can be independently scheduled with only a small additive effect on the number of rounds (as shown in Figure 5).

Another possible concern with this optimization is that, instead of a single message front outgoing from the sender and then returning from the farthest node, we now have *multiple* message fronts traversing the network simultaneously. Since the applications that drive the linear topology are primarily wireless, interference and medium contention between may prevent nodes that are close together from transmitting simultaneously. However, we can show that for a fixed interference radius δ, this implies at most an additional δ rounds. We perform the following adaptations: instead of splitting the receiver set into $\log n$ subsets, we stop halving the sets when the smallest subset still has size at least δ; this means that the receiver sets are of size $n/2, n/4, \ldots$ and so on until the smallest set has size between $2\delta - 1$ and δ. We then run the algorithm as normal, except that the returning passes do not start until the outgoing pass has proceeded out of interference range, i.e., we delay each returning pass by δ rounds. It is not hard to see that this modification completes within $n + \delta$ rounds and the communication congestions remains bounded by $O(\log n)$. Hence, limited-radius interference only contributes a fixed additive effect to the number of rounds and does not fundamentally affect the round complexity of the protocol.

B. Hash Chain Based Schemes

The operation of hash-chain based broadcast authentication schemes is described in Section IV-B. Within the family of hash chain schemes, TESLA [9] uses time synchronization to ensure all receivers have received the MAC before releasing the hash chain key. Not only is secure time synchronization an additional complication, but furthermore, to ensure that a legitimate message is not rejected simply because it was delayed in transit, the delay between broadcasting the MAC and releasing the key must be at least the worst-case estimate of the round complexity from the sender to the furthest node, which may be many times the average case round complexity. In contrast, Guy-Fawkes schemes like CSA [7] use explicit acknowledgments sent from the receivers to the sender; only when the sender has received acknowledgments from all nodes does it release the hash chain key. This results in a three-phase protocol with a round complexity of $3n$ rounds. We now describe two optimizations that combine to reduce the number of passes of the protocol from 3 to 2 passes, allowing us to use the technique in Section V-A2 to improve this to optimal n round complexity.

1) Optimization 1: Use Aggregated Acknowledgments: In CSA, each receiver transmits a separate hash-chain based ACK message back to the sender. This causes n messages in total to be received by the sender, thus the communication congestion of the scheme is n cryptographic values. Yao et al. improve this for tree-based networks by proposing the construction of an authenticated aggregate acknowledgment via construction of a hash tree over the acknowledgments [5], which reduces the congestion to a single hash value; however the sender must know the exact topology of the tree to verify the aggregated acknowledgment (instead of just the receiver set).

We propose using XOR-based aggregate acknowledgments [28], [2], [29] which can provide equivalent functionality without these drawbacks. The construction can be summarized as follows: to acknowledge receipt of a message M_j with MAC T_j, receiver d_i releases the value $\mathrm{PRF}_{K_i}(T_j \| M_j)$ to the network where K_i is the secret key shared between receiver d_i and the sender, and PRF is a cryptographic hash function. Since K_i is secret, no other node except receiver i can produce this value. These values are aggregated using XOR as they traverse the return-path to the sender until ultimately the sender receives $A_j = \bigoplus_{i=1}^{n} \mathrm{PRF}_{K_i}(T_j \| M_j)$. Since the sender knows all the K_i, it can then check the received aggregate value to see if it is correctly the XOR of all the PRF values computed by each receiver; if any of the receivers did not supply the correct PRF value then the received aggregate value will be incorrect. Katz and

Lindell formally proved that the aggregate MAC is secure against existential forgery if each individual PRF component is also secure against existential forgery [29]. By using this aggregated acknowledgment in the acknowledgment phase of Guy Fawkes, we can achieve a 3-pass protocol with constant congestion.

2) Optimization 2: Encrypt the Hash Chain Key: In Guy Fawkes, the third pass where the hash chain key h_j is disseminated starts only after the sender has confirmed that all receivers have received the MAC. When using XOR-aggregated acknowledgments, this involves checking the received aggregate acknowledgment against the correct value A_j that is the XOR of all the expected PRF values. We make two observations. First, computation of the aggregate acknowledgment can be performed on the initial outgoing pass; this is because, similarly to the case for the hash-tree schemes, it is safe for receiver d_i to release its value $\text{PRF}_{K_i}(T_j \| M_j)$ as soon as it has received message M_j. This allows each node to compute the running XOR of its own PRF value with the values of all the preceding receivers, until the final receiver d_n computes $A_j = \bigoplus_{i=1}^{n} \text{PRF}_{K_i}(T_j \| M_j)$. Second, we note that (1) A_j can be precomputed by the sender and (2) A_j remains computationally indistinguishable from a random value to the adversary as long as it contains a PRF value from at least one legitimate receiver. Hence, A_j can be used as a *key* to encrypt the value of h_j; since A_j is only revealed once all legitimate receivers have received M_j, this is precisely the time to expose h_j. In particular, in a linear topology, this sender can include $E_j = E_{A_j}(h_j)$ in the broadcast, where $E_K(X)$ denotes the encryption of string X with key K. When the broadcast reaches the last node d_n, it then decrypts E_j to reveal h_j which is then sent back upstream to allow the rest of the receivers to authenticate the message. The result is a protocol with only two passes.

In general E can be any encryption function (e.g., AES), but since each A_j is used as a key to encrypt only one fixed-length plaintext h_j, if we set the output lengths of the PRF and the hash function to be equal, we can simply use A_j as a pseudorandom masking factor for h_j, i.e., $E_j = [\bigoplus_{i=1}^{n} \text{PRF}_{K_i}(T_j \| M_j)] \oplus h_j$. From this construction, it is clear that if even one PRF value is unknown, the adversary finds it infeasible to recover any bits of h_j from E_j. The protocol is described in pseudocode in Algorithm 3; the proof of the security of this optimization is a straightforward reduction of attacking the protocol to attacking either the pre-image resistance of H or the distinguishability of the PRF from a random function [26].

Like the basic CSA scheme, the algorithm requires a freshness check on M_j. This prevents the adversary from using possible old revealed values on the hash chain to authenticate bogus messages.

The resultant protocol terminates in 2 passes and thus results in a round complexity of $2n-1$ for a linear topology.

Algorithm 3 Hash Chain Protocol with 2 passes

```
Require: Hash chain $h_0, \ldots, h_m$
  $M_j = j$th Message
  $s$ computes: $T_j = \text{MAC}_{h_j}(M_j)$
  $s$ computes: $E_j = \bigoplus_{i=1}^{n} \text{PRF}_{K_i}(T_j \| M_j) \oplus h_j$
  $A_j \leftarrow 0$
  $s$ sends $(M_j, T_j, E_j, A_j)$ to $d_1$:
  for $i$ = 1 to $n$ do
    Check that $M_j$ is from stage $j$;
    If $d_i$ has already attempted the algorithm for stage $j$ or
    later, abort.
    $d_i$ computes: $A_j \leftarrow A_j \oplus \text{PRF}_{K_i}(T_j \| M_j)$
    $d_i$ forwards $(M_j, T_j, E_j, A_j)$ to $d_{i+1}$
  end for
  $d_n$ computes: $h_j \leftarrow A_j \oplus E_j$
  for $i = n$ downto 1 do
    $d_i$ checks $T_j = \text{MAC}_{h_j}(M_j)$; if success, accept $M_j$.
    $d_i$ forwards $h_j$ to $d_{i-1}$
  end for
```

Each of T_j, E_j, A_j and h_j are passed through each node in the network, resulting in a total of 4 cryptographic values are sent through each receiver in the network. The total communication cost is thus $4n$ cryptographic values. The maximum per-round data transfer increases from 1 cryptographic value to 3.

3) Partitioning the receiver set: Since this protocol is now a 2-pass protocol, the speedup trade-offs of Section V-A2 apply directly. In general, $n + n/(2^k - 1)$ round complexity is achievable with k subsets implying a congestion overhead of $2(k+1)$ cryptographic values. Total communication cost remains n times the congestion, i.e., $2n(k+1)$. The maximum per-round data transfer increases to $2k+1$ cryptographic values. Optimal n round complexity is achievable with $k = \lceil \log n \rceil + 1$ receiver sets, at the cost of $2(\lceil \log n \rceil + 2)$ communication congestion. Total communication cost is n times the congestion, i.e., $2n(\lceil \log n \rceil + 2)$. The maximum per-round data transfer increases to $2\lceil \log n \rceil + 3$ cryptographic values; however, as discussed in Section V-A2, this is unlikely to be a limiting factor.

VI. Fully Connected Topology

We now examine bounds and optimizations for the fully connected topology. In the fully connected topology, G is the complete graph K_{n+1} of $n+1$ vertices (including the sender s and n receivers).

Fully-connected topologies are important in both theory and practice. In terms of theoretical importance, round complexity lower bounds proven on fully connected topologies apply to *all* topologies. Since any graph G of $n+1$ nodes is a subgraph of K_{n+1}, any algorithm for G is also an algorithm for K_{n+1}; hence if we can show that no algorithm can be

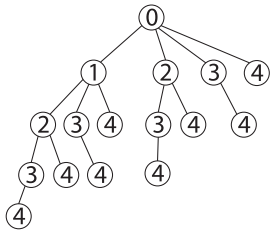
Figure 6. Maximum Dissemination Tree

faster than $f(n)$ on the complete graph K_{n+1}, the bound also applies directly to arbitrary topologies. In terms of practical importance, fully-connected topologies occur whenever a network of nodes can achieve any-to-any addressability; this commonly occurs on Internet-based distributed systems such as peer-to-peer networks and grid computing.

A. Doubling Broadcast / Maximum Dissemination Trees

Most of the algorithms in this and subsequent sections rely on well-known techniques and bounds for fast information dissemination in arbitrary networks. Hence, as a preliminary, we first consider the problem of disseminating a message M from one node to as many other nodes as possible within t rounds. A straightforward approach is *doubling broadcast*: in each round, every node which has knowledge of M communicates it to a different node that has not yet received M. An inductive reasoning on t shows that this schedule reaches the maximum number of nodes for each t; since each node can only communicate with one other node in each round, the number of nodes that have received M can at most double at each round. Thus, after t rounds, up to 2^t nodes have received M. Figure 6 shows this process; the numbers in the circled nodes indicate in which round the node first receives the message. We define the communication pattern caused by this optimal dissemination schedule the *maximum dissemination tree*. A consequence of this schedule being optimally fast is that any dissemination of a message from one node to n nodes must take at least $\lceil \log_2(n) \rceil$ rounds on any topology.

B. Hash Tree Based Schemes

As mentioned in Section IV-A, the hash tree based broadcast authentication scheme takes three message passes to and from the sender to the rest of the network. Since the network is fully connected, it is easy to derive a maximum dissemination spanning tree rooted at the sender that reaches all nodes. The tree will have depth $\lceil \log(n+1) \rceil$; three passes on this tree will take $3\lceil \log(n+1) \rceil$ rounds; this is the fastest possible round complexity for the unoptimized protocol. Authentication paths on this hash tree will have $O(\log n)$ length; congestion and max per round data transfer are also $O(\log n)$.

As an improvement, we present a schedule which takes at most $2\log n + 1$ rounds, with a maximum per-round

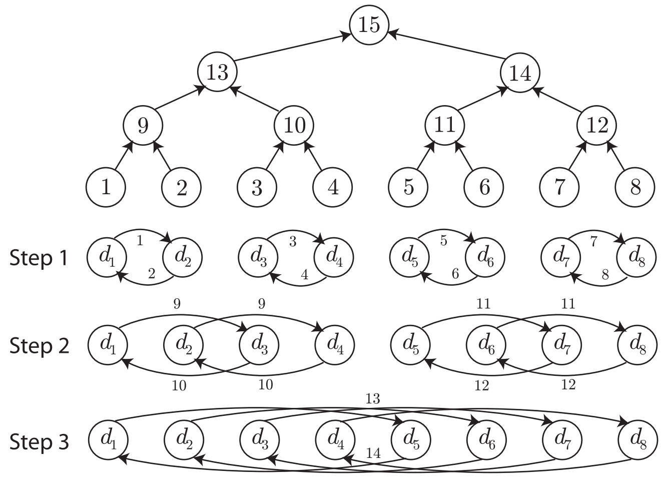

Figure 7. Path Reconstruction Phase

data transfer of 2 cryptographic values. We first describe an algorithm for the case where n is a power of 2, i.e., the hash tree is a perfect binary hash tree.

The protocol proceeds in two phases. In the first phase, the sender computes the hash tree root r for the message M using the construction of Section IV-A. Then the message tuple (M, r) is disseminated using the repeated maximum dissemination broadcast method of Section VI-A. This takes $\log n + 1$ rounds: one round for the sender to send the initial message, then $\log n$ rounds for all receivers to receive it. In the second phase, nodes collaborate to reconstruct authentication paths. This phase of messages is illustrated in Figure 7. There is one round of message exchanges for each level of the hash tree, starting from the leaves. At each level j, every node exchanges the hash tree vertex at level j on its authentication path with that of its counterpart in the neighboring subtree at level j. This allows it to compute the next higher hash tree vertex (at level $h-1$) on its authentication path and thus repeat the process in the next round. Within $\log n$ rounds all nodes will have computed their authentication paths to the root. This process is shown in Algorithm 4. The communication pattern is similar to all-to-all broadcast on a hypercube.

It remains to address the case where n is not a power of 2. In this case, we can partition the receiver set into up to $\log n$ subsets where each subset is a distinct power of 2. The subsets correspond to the binary representation of n, e.g., for $n = 13$, we have subsets of size $8, 4$ and 1. The protocol then runs, in parallel, one completely separate instance of the protocol for each subset. Since the sender can only start one instance of the protocol in each round, it starts the protocol for the largest subset first, then it starts the protocol for the second largest subset in the next round, and so on. Even with this head start, the largest subset has the largest hash tree and will be the last subset to complete verification; it has $2^{\lfloor \log n \rfloor}$ nodes and thus takes $1 + 2\lfloor \log n \rfloor \leq 2\log n + 1$

Algorithm 4 Hash Tree Scheme

```
$s$ sends $(M, r)$ to all receivers using doubling broadcast (Sec. VI-A).
Check freshness of $M$ (e.g., inspect seq. num), if not fresh, abort.
Otherwise, $d_i$ sets $v_i[0] \leftarrow \text{PRF}_{K_i}(M)$
for $h$ = 0 to $(\log n) - 1$ do
  for $i$ = 1 to $n$ do
    $j \leftarrow ((i-1) \oplus 2^h) + 1$
    $d_i$ sends to $d_j$: $v_i[h]$
    $d_i$ receives from $d_j$: $v_j[h]$
    if $i < j$ do $v_i[h+1] \leftarrow H(v_i[h] \| v_j[h])$
    if $i > j$ do $v_i[h+1] \leftarrow H(v_j[h] \| v_i[h])$
  end for
end for
Each $d_i$ checks authentication path in $v_i$.
```

rounds to complete verification.

The greatest amount of per-round data transfer occurs when the nodes are exchanging hash-tree vertices: since one hash value is transmitted and received respectively, a total of 2 cryptographic values are exchanged. The communication congestion overhead for this scheme is at most $2\lceil \log n \rceil$ values: $\lceil \log n \rceil$ for the dissemination of r, and $\lceil \log n \rceil$ for the authentication path. The sender is responsible for sending up to $\lceil \log n \rceil$ cryptographic values, leading to a total communication overhead of $(2n+1)(\lceil \log n \rceil)$.

Hash Chain Algorithms: We note that a 2-pass hash-chain based scheme (see Section V-B2) can follow the same schedule, resulting in similar round complexity and communication overheads. Instead of exchanging hash tree vertices, receivers will exchange their XOR-aggregated acknowledgments; by the end of the protocol each receiver has collected the aggregated acknowledgments of all the receivers in the network and can thus decrypt the hash chain value and authenticate the broadcast message.

C. Lower Bounds on Round Complexity vs Congestion

Several of our schemes for the linear topology are round complexity-optimal: they take n rounds, which is the lower bound on the number of rounds needed for message dissemination. It is less clear what lower bounds exist for the fully connected topology. The requirements for message dissemination ($\log n$ rounds for doubling broadcast) provide a loose bound, which can be achieved by attaching a digital signature to every message. However digital signatures represent a fundamentally harder problem than authentication, being both more computationally expensive and providing additional unnecessarily strong properties such as non-repudiation. Therefore, we investigate what bounds can be derived for protocols which do not involve the digital signature problem.

We prove that, any network broadcast authentication protocol that with a fixed communication pattern that completes in at most $(2-\rho)\log n + k$ rounds (where k is a constant) must imply a digital signature or have communication overhead at least proportional to n^ρ. By implication, any protocol that completes in asymptotically less than $2 \log n$ rounds cannot have communication complexity polylogarithmic in n unless it also provides signature properties (i.e., it solves a fundamentally harder problem).

We build on the result of Boneh, Durfee and Franklin [18] which showed that any *multicast MAC* (or MMAC) for n receivers must either imply a digital signature or have overhead proportional to n. An MMAC is defined by three algorithms KEY-GEN, MAC-GEN, MAC-VER. Algorithm KEY-GEN takes a security parameter s and a number of receivers n and generates the sender key **sk** and receiver keys $\mathbf{rk}_1, \ldots, \mathbf{rk}_n$ for each receiver $d_1, \ldots, d_n$. Algorithm MAC-GEN takes a payload message M and a sender key **sk** returns a fixed-length tag $T = \text{MAC-GEN}(M, \mathbf{sk})$. Algorithm MAC-VER takes a payload message M, a tag T, and a receiver key K and returns a bit indicating whether the tag T verified correctly for payload message M, under the constraint that correctly generated tags from the sender always verify successfully for all receivers, i.e., $\text{MAC-VER}(M, \text{MAC-GEN}(M, \mathbf{sk}), \mathbf{rk}_i) =$ *'accept'* for all $i = 1, \ldots, n$.

MMACs are static codes; in this sense they can be considered a special case of the set of broadcast authentication protocols under the restriction that the sender only produces a single, fixed-length tag T for all receivers. Each receiver must then perform verification by inspection of T without interacting with the sender or other receivers. General broadcast authentication protocols, on the other hand, allow the sender to send different protocol messages to different receivers, which can perform rounds of interaction with each other or the sender to determine if the payload is legitimate. Boneh, Durfee and Franklin's result holds only for MMACs and do not apply to broadcast authentication protocols in general. We extend their result to show that if a broadcast authentication protocol is fast, then it implies an MMAC. The basic idea is that if the protocol is fast, then there must exist a (large) set of receivers that did not interact with each other in the protocol; in the absence of interaction between receivers, a broadcast authentication protocol can be reduced to an MMAC.

The details of the proof are as follows. To begin with, assume that all receivers are deterministic, i.e., they do not perform any purely random coin tosses during the protocol, although they may simulate this with pseudorandom values generated from a preloaded seed. We assume that the communication pattern is static for each execution of the protocol, i.e., the set of nodes that communicate with each other in each round is fixed regardless of the message or tag originated by the sender. These assumptions are consistent with all known broadcast authentication protocols in the literature. We do not count setup overhead (such as key establishment, etc) in the rounds of the protocol; thus we can assume that no receiver starts the protocol until it receives a message that contains information from the sender.

The proof involves a reduction from a fast broadcast authentication protocol to an MMAC; we define the security games for each class of problems as follows. For the MMAC security game: (1) The adversary *adaptively* selects a proper subset S_A of S for compromise, and the challenger provides A with all secret information known by these receivers as they are selected; the adversary can expand her choice of S_A based on this information. (2) The adversary can query the challenger to provide valid MMACs for some adaptively chosen $M_1, \ldots, M_q$. (3) The adversary constructs a forged data-message $M' \neq M_i$ for all $i, \ldots, q$ and a tag T'. The adversary wins the game if some receiver in $S - S_A$ accepts message M'. We say the MMAC is ϵ-secure if the probability of any adversary winning is at most ϵ.

The security game for existential forgery on the broadcast authentication protocol is similarly structured: (1) The adversary adaptively selects a subset of receivers C for compromise, and the challenger provides all secret information known by these receivers. (2) The adversary can query the challenger to execute the authentication protocol for some adaptively chosen $M_1, \ldots, M_q$. (3) The adversary constructs a forged data-message $M' \neq M_i$ for all $i, \ldots, q$, and sends a (polynomially bounded) series of messages to some receivers not in C. The adversary wins if any of these uncompromised receivers accept M' because of these messages. We say the protocol is ϵ-secure if the probability of any adversary winning is at most ϵ.

The construction of a MMAC from a broadcast protocol is based on finding a set of nodes that do not interact directly or indirectly in the protocol. More formally:

Definition 1. *For a given sequence of messages in a protocol, Node A is an* upstream *node of node B if A could have sent any information to Node B in the protocol. Let B be a* downstream *node of node A iff node A is upstream of B.*

More precisely, A is upstream of B if there exists a sequence of nodes $X_1, \ldots, X_k$ and messages $m_1, \ldots, m_k$ in successive (not necessarily consecutive) rounds that start with A and end with B (e.g., $A \rightarrow X_1 : m_1$, $X_1 \rightarrow X_2 : m_2$, ..., $X_k \rightarrow B : m_k$ in successive rounds).

Lemma 1. *Let P be a broadcast authentication protocol that is ϵ-secure against arbitrary coalitions of receivers that has congestion at most $c(n)$ bits. Let S be a subset of m receivers in the protocol P such that no two members of S are upstream (or downstream) of each other. Then there exists an MMAC for m receivers with tag length no more than $c(n)$ bits that is also ϵ-secure.*

Proof: We can construct a MMAC for receivers $s_1, \ldots, s_m$ in S by defining the three subroutines KEY-GEN,MAC-GEN and MAC-VER in terms of the broadcast authentication protocol P. Let KEY-GEN be defined as follows: for each receiver s_i in S, let $\mathbf{rk}_i$ be the receiver key of s_i in P; let the information **sk** known to the sender be all the information known by all principals in the broadcast protocol. This allows the sender to fully simulate the execution of any run of the protocol; in particular it can predict *a-priori* the set of messages it will send in any run of the protocol for a given message M. Hence, let the tag T generated by MAC-GEN$(M, \mathbf{sk})$ be the ordered set of all messages sent by the sender for a given broadcast of message M. We define MAC-VER$(M, T, \mathbf{rk}_i)$ such that it simulates the operation of P to the point of verification by receiver s_i. Let R_S be the set of all nodes that are upstream of any node in S. By definition of S, $R_S \cap S = \emptyset$. Encode (publicly) inside MAC-VER all the protocol information associated with the nodes in R_S (including their receiver keys). By definition of R_S, this information allows any MMAC receiver s_i to accurately simulate the entire broadcast protocol from the point of view of node s_i in the broadcast protocol; in other words, all messages sent and received by node s_i in the broadcast protocol can be generated from this information by MMAC receiver s_i. We can thus define MAC-VER$(M, T, \mathbf{rk}_i)$ to return *'accept'* if s_i accepts the payload M in the simulation of the protocol P where the sender sends the messages encoded in T, and *'reject'* if s_i does not accept M.

Since the protocol P has at most $c(n)$ bits of congestion, the tag T is no more than $c(n)$ bits since it consists of all messages from a single node (the sender). It remains to show that the MMAC construction is at least as secure as P.

Let adversary A be an adversary for the MMAC game that wins with probability ϵ. We define Adversary B for attacking the broadcast authentication protocol that runs A as a subroutine. Adversary B first compromises R_S, the set of all nodes upstream of S. From this, B constructs the KEY-GEN,MAC-GEN and MAC-VER for the MMAC, which is fed to A. Now A adaptively selects some set S_A for compromise; these nodes are also adaptively compromised by B in the broadcast authentication protocol. By definition of S, $R_S \cap S = \emptyset$, so the set $S - S_A$ remain uncompromised for both adversaries. During the chosen message attack phase, queries of A are forwarded to B's challenger, and the corresponding MMACs are reconstructed by inspection of the sender's messages for M_i. Now when attacker A produces M' and T', since Adversary B has compromised all receivers upstream of S, it can use this information as a script to (interactively and independently) simulate the protocol P to each member of S. Since MAC-VER(M', T', rk_i) succeeds with at least ϵ probability for some receiver in $S - S_A$, by construction, the chance of Adversary B successfully causing a receiver in $S - S_A$ to accept the forged broadcast message M' is also ϵ. ■

Lemma 2. *In any broadcast authentication protocol, for any topology, at least $n/2$ nodes start executing the protocol only after $\lfloor \log n \rfloor$ rounds.*

Proof: The doubling broadcast of Sec. VI-A) is the fastest rate at which information can spread in the network. At round $\lfloor \log n \rfloor - 1$, no more than $2^{\lfloor \log n \rfloor - 1} \leq n/2$ nodes have received their first message of the protocol; the remaining nodes must start the protocol after $\lfloor \log n \rfloor$ rounds. ■

Lemma 3. *Let G be a network topology and $L \subseteq V$ be a subset of the nodes in G. Let $U(v,t)$ be an upper bound on the number of nodes in L that can receive information from a node $v \in L$ within t rounds. Then $U(v,t)$ is also an upper bound on the number of nodes in L that can send information to v within t rounds in the topology.*

Proof: Any schedule for disseminating information from v can be converted to a schedule for converge-casting information to v by reversing the schedule chronologically and reversing the direction of all communications. Hence, in a fixed topology, the maximum possible number of upstream and downstream nodes from a given node v are the same. ■

Lemma 4. *If a broadcast authentication protocol takes no more than $(2-\rho)\lfloor \log n \rfloor$ rounds, then there exists a set S of at least $\frac{1}{4}n^{\rho}$ receivers that are not mutually upstream or downstream of each other.*

Proof: Let L be the set of receivers which started the protocol after round $\lfloor \log n \rfloor$. By Lemma 2, $|L| \geq n/2$. The nodes in L only have $(1-\rho)\lfloor \log n \rfloor$ rounds remaining in which to exchange information. By the doubling broadcast of Section VI-A, any node in L can reach at most $2^{(1-\rho)\log n} \leq n^{1-\rho}$ other members of L within that time; by Lemma 3, likewise less than $n^{1-\rho}$ of the other nodes in L can be upstream nodes of l_i. Hence, we can construct S as follows: starting with empty S, select any node l_i in L; remove it from L and add it to S; also remove all nodes in L that are upstream or downstream of l_i. After each selection, the size of L decreases by less than $2n^{1-\rho}$ nodes. Hence, we can perform such a selection at least $\frac{1}{2}n/2n^{1-\rho} = \frac{1}{4}n^{\rho}$ times. The resultant S has at least $\frac{1}{4}n^{\rho}$ nodes each of which are neither upstream nor downstream of each other. ■

Theorem 1. *Any deterministic ϵ-secure broadcast protocol taking at most $(2-\rho)\lfloor \log n \rfloor$ rounds either implies an $(\epsilon + 1/2^m)$-secure signature mechanism or requires at least $\frac{1}{4}n^{\rho} - m$ bits of congestion where m is a constant security parameter (e.g., 128).*

Proof: By Lemmas 4 and 1, any deterministic broadcast protocol taking at most $(2-\rho)\lfloor \log n \rfloor$ rounds implies an equivalently-secure MMAC for $\frac{1}{4}n^{\rho}$ receivers with tag length equal to the congestion at the sender. Boneh, Durfee and Franklin [18] showed that an ϵ-secure MMAC for r receivers with tag length no more than $r - m$ bits implies an $(\epsilon + 1/2^m)$-secure signature scheme. Hence, the broadcast protocol either involves the use of a $(\epsilon + 1/2^m)$-secure signature primitive, or it must incur at least $\frac{1}{4}n^{\rho} - m$ bits of congestion at the sender. ■

More significantly, this implies that no deterministic broadcast protocol can take asymptotically less than $2\log n$ rounds while achieving congestion complexity polylogarithmic in n. This means that the hash tree based algorithm of Section VI-B achieving $2\log n + 1$ rounds is asymptotically optimal for protocols with polylog congestion complexity; in other words the existence of the algorithm of Section VI-B shows that bound we have proved is tight for polylog-congestion algorithms under the full-duplex communication model. As noted earlier in the beginning of SectionVI, this lower bound holds for all topologies. Since the algorithm of Section VI-B relies on a two-way exchange of information in each round, whether this bound is tight for the half-duplex communication model (where all communications are strictly one-way in each round) remains an open question.

VII. Extending the Lower Bound to Tree Topologies

The method of proving the round complexity lower bound of $2\log n$ for arbitrary topologies can be extended to trees. The proof for trees follows the same structure as the proof for fully connected topologies (Section VI-C); while information spreads at the rate of 2^t in fully-connected topologies, we can show that it spreads at a slower rate between leaf nodes in trees and hence the round complexity lower bound is higher. Specifically, we prove the following lemma:

Lemma 5. *Let v be a vertex at the* greatest depth *in an arbitrary tree topology with a designated root r. The number of nodes that can be reached from v within t rounds is no more than $\phi^{t+3}/\sqrt{5}$, where ϕ is the golden ratio (≈ 1.618).*

Proof: We consider the set of all nodes reachable from each node at each depth from u to the root. We define a term $T(t,h)$ as the size of the h *height-bounded maximum dissemination tree* (see Section VI-A, Figure 6) of t rounds. This can be defined as the maximum size of the set of nodes reached from a single node v_0 given t rounds such that no node is more than h hops away from v_0. This can be obtained from the maximum dissemination tree $T(t)$ by truncating off all levels with depth greater than h.

We observe that $T(t,h) = T(t-1,h) + T(t-1,h-1)$. This follows from the observation that in the first round, the original node v_0 can send to at most one other node v_1; in the remaining $t-1$ rounds, each of these two nodes then continue disseminating into their respective subtrees, except that the height bound of the maximum dissemination

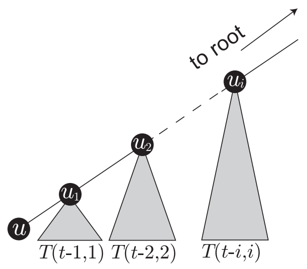

Figure 8. Reachable nodes

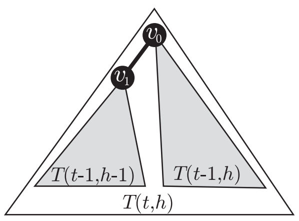

Figure 9. Structure of $T(t,h)$

tree rooted at v_1 is one less than the original height bound h. This reasoning is shown in Figure 9. For base cases, we note that, for any x, $T(x,0)=1$ (a height bound of 0 does not allow the original node to disseminate to any other node) and $T(0,x)=1$ (with 0 rounds, no dissemination takes place).

We can now consider the maximum number of nodes reachable from a max-depth leaf vertex u within t rounds. The argument is illustrated in Figure 8. The figure shows the nodes between u and the root r. The node u_i is i levels above u. It can be reached no earlier than at round i. Thus, u_i has $t-i$ rounds remaining to disseminate into its subtree. Since u is a node at maximum depth in the tree, the subtree rooted at u_i has a height bound of i. Hence, the maximum number of nodes reached from u_i is $T(t-i,i)$. A bound on the total number of nodes reachable from u in t rounds is thus $F(t)=\sum_{i=0}^{t}T(t-i,i)$. Using the recurrence $T(t,h)=T(t-1,h)+T(t-1,h-1)$ we can expand $F(t)$ as follows:

$$\begin{aligned}
F(t) &= \sum_{i=0}^{t} T(t-i,i)\\
&= T(t,0)+\sum_{i=1}^{t-1} T(t-i,i)+T(0,t)\\
&= 1+\sum_{i=1}^{t-1}\left[T(t-i-1,i)+T(t-i-1,i-1)\right]+1\\
&= T(t-1,0)+\sum_{i=1}^{t-1} T(t-1-i,i)\\
&\quad +\sum_{i=1}^{t-1} T(t-2-(i-1),i-1)+1\\
&= \sum_{i=0}^{t-1} T(t-1-i,i)+\sum_{j=0}^{t-2} T(t-2-j,j)+1\\
&= F(t-1)+F(t-2)+1
\end{aligned}$$

By substituting $F'(t)=F(t)+1$, we can see that $F'(t)=F'(t-1)+F'(t-2)$ yielding a Fibonacci series. From the base cases $F(0)=1, F(1)=2$ we can show $F(t)=\mathrm{Fib}(t+3)-1$ where $\mathrm{Fib}(x)$ is the xth number in the standard Fibonacci series. From Binet's formula, $\mathrm{Fib}(x)\le\lceil\phi^x/\sqrt{5}\rceil$, hence $F(t)<\phi^{t+3}/\sqrt{5}$. ■

Lemma 6. *If a broadcast authentication protocol on a tree topology takes no more than $(2.44-\rho)\log n-2.33$ rounds, then there exists a set S of at least $\frac{1}{4}n^{0.694\rho}$ receivers that are not mutually upstream or downstream of each other.*

Proof: Let L be the set of receivers which started the protocol after round $\log n-1$. By Lemma 2, $|L|\ge n/2$. Consider the maximum-depth node u in L relative to the sender at the root of the tree topology. We can bound the number of members of L reachable by u in the remaining rounds after the first $\log n$ rounds by Lemma 5 because any nodes that are at a greater depth than u in the tree are all not in L and are hence irrelevant to the count. Let t be the number of rounds needed for u to reach all n receivers. We can set $\phi^{t+3}/\sqrt{5}=n$, and solve $t=\log_\phi n+(\log_\phi\sqrt{5}-3)>1.44\log n-1.33$. Hence, in the remaining $(1.44-\rho)\log n-1.33$ rounds, u can reach at most $\phi^{(1.44-\rho)\log n+1.67}/\sqrt{5}\le n^{1-0.694\rho}$ total members of L (including u itself); by Lemma 3, likewise less than $n^{1-0.694\rho}$ of the other nodes in L can be upstream nodes of l_i. Hence, we can construct S as follows: starting with empty S, select a node l_i that is at the greatest depth in the tree of all the nodes in L; remove it from L and add it to S; also remove all nodes in L that are upstream or downstream of l_i. Repeat until L is empty. After each selection, the size of L decreases by no more than $2n^{1-0.694\rho}$ nodes. Hence, we can perform such a selection at least $\frac{1}{2}n/2n^{1-0.694\rho}=\frac{1}{4}n^{0.694\rho}$ times. The resultant S has at least $\frac{1}{4}n^{0.694\rho}$ nodes each of which are neither upstream nor downstream of each other. ■

Theorem 2. *Any deterministic ϵ-secure broadcast protocol on a tree topology taking at most $(2.44-\rho)\log n-2.33$ rounds either implies an $(\epsilon+1/2^m)$-secure signature mechanism or requires at least $\frac{1}{4}n^{0.694\rho}-m$ bits of congestion where m is a constant security parameter (e.g., 128).*

Proof: Identical to the proof of Theorem 1, except using Lemma 6 instead of Lemma 4. ■

Theorem 2 implies that any protocol with at most $(2.44-\rho)\log n-2.33$ rounds on a tree topology must involve either a signature scheme or congestion polynomial in n. In particular this rules out the possibility of strictly 2-pass schemes on max-dissemination trees with polylog congestion, since such a scheme would take only $2\log n$ rounds. From the discussion at the beginning of Section VI-B, we note that a 3-pass broadcast authentication protocol takes only $3\log n$ rounds on maximum dissemination trees; we conjecture that the small gap factor between 2.44 and 3 seems to indicate that the current 3-pass schemes are likely round complexity-optimal for such trees, i.e., the true lower bound over all trees is possibly $3\log n$.

	Rounds	Max data per round	Congestion	Total comm.
Linear Topology				
HT (Basic) [2]	$3n-2$	$\lceil\log n\rceil$	$\lceil\log n\rceil+2$	$n\lceil\log n\rceil$
HT (Sec.V-A1)	$2n-1$	$\lceil\log n\rceil+1$	$\lceil\log n\rceil+2$	$n\lceil\log n\rceil$
HT, k subsets (Sec.V-A2)	$n+\frac{n}{(2^k-1)}$	$\lceil\log n\rceil+k+1$	$\lceil\log n\rceil+k+1$	$n(\lceil\log n\rceil+k)$
HT, $\lceil\log n\rceil+1$ subsets	n	$2(\lceil\log n\rceil+1)$	$2(\lceil\log n\rceil+1)$	$2n(\lceil\log n\rceil+1)$
HC (Basic) [5]	$3n$	1	3	$3n$
HC, 2-pass (Sec.V-B2)	$2n-1$	3	4	$4n$
HC, k subsets (Sec.V-B2)	$n+\frac{n}{(2^k-1)}$	$2k+1$	$2(k+1)$	$2n(k+1)$
HC, $\lceil\log n\rceil+1$ subsets	n	$2\lceil\log n\rceil+3$	$2(\lceil\log n\rceil+2)$	$2n(\lceil\log n\rceil+2)$
Tree Topology				
Lower Bound (Sec.VII)	$(2.44-\rho)\log n-2.33$	–	$\Omega(n^{0.694\rho})$	–
Full Topology				
HT (Max. Dissem. Tree)	$3\lceil\log(n+1)\rceil$	$\Theta(\log^2 n)$	$\Theta(\log^2 n)$	$\Theta(n\log^2 n)$
HC (Max. Dissem. Tree)	$3\lceil\log(n+1)\rceil$	1	$\lceil\log n\rceil$	$3(n+1)$
Lower Bound (Sec.VI-C)	$(2-\rho)\log n$	–	$\Omega(n^\rho)$	–
HT/HC (Sec.VI-B)	$2\lceil\log n\rceil+1$	2	$2\lceil\log n\rceil$	$(2n+1)\lceil\log n\rceil$

HT: Hash Tree based scheme; HC: Hash Chain based scheme.
Italics denote unoptimized schemes for comparison.
Lower Bounds establish a relationship between min. number of rounds and min. congestion

Table I
ROUND COMPLEXITY AND CONGESTION BOUNDS OF OUR PROTOCOLS.

VIII. Summary and Conclusion

A summary of the results presented in this paper is presented in Table I. The new protocols avoid the high computation overhead of digital signatures and the high communication overhead of one-time signatures and multi-receiver MACs, as well as the time synchronization needed by TESLA. In terms of round complexity and communication congestion, our protocols provide points in the design space that are not achievable by previously published protocols. In particular, for the linear topology, the hash tree scheme achieves the fastest possible round complexity n with only $O(\log n)$ congestion; the hash chain scheme achieves constant congestion with only $2n$ round complexity. For the fully connected topology, the hash tree scheme requires at most $2\lceil\log n\rceil+1$ round complexity with only $O(\log n)$ congestion; since we also show that this is the fastest possible round complexity achievable for schemes with polylogarithmic congestion, this also functions as an existence proof that the $2\log n$ bound is asymptotically tight on the coefficient of $\log n$ for the full-duplex communication model. Our optimizations do not significantly increase the per-round communication cost.

The other main contribution of this paper is in showing previously uninvestigated round complexity bounds for symmetric-key-based broadcast authentication in fully connected and tree topologies. We show a tight round complexity bound of $2\log n$ rounds for broadcast authentication schemes with polylogarithmic congestion in any topology and another round complexity lower bound of $2.44\log n-2.33$ rounds for trees. These lower bounds show the fundamental round complexity limitations of broadcast authentication protocols, and can inform the design constraints of new authentication protocols in networks. The new bounds also significantly extend the relevance of the result on multicast message authentication by Boneh, Durfee and Franklin [18]. The previous result applied only to static *authentication codes* and did not address *protocols* involving multiple interactions between receivers, which are in fact the primary methods of multicast message authentication in practice. In our proofs of round complexity lower bounds, we greatly improve the relevance of this result by showing how it can be extended to apply to general authentication protocols by considering round complexity. This significantly expands the existing level of understanding about the fundamental limits in signature-free multicast authentication.

Acknowledgments

We would like to thank Patrick Tague and the anonymous reviewers for their helpful feedback. This research was supported by CyLab at Carnegie Mellon under grants DAAD19-02-1-0389 and MURI W 911 NF 0710287 from the Army Research Office, and grant CNS-0347807 from the National Science Foundation. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of ARO, CMU, CyLab, NSF, or the U.S. Government or any of its agencies.

References

[1] R. Anderson, F. Bergadano, B. Crispo, J. Lee, C. Manifavas, and R. Needham, "A new family of authentication protocols," *SIGOPS Oper. Syst. Rev.*, vol. 32, no. 4, pp. 9–20, 1998.

[2] H. Chan and A. Perrig, "Efficient security primitives derived from a secure aggregation algorithm," in *Proc. ACM Conf. on Computer and Communications Security*, 2008, pp. 521–534.

[3] S. Johnsson and C. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Trans. on Computers*, vol. 38, no. 9, pp. 1249–1268, 1989.

[4] N. Lynch, *Distributed algorithms*. Morgan Kaufmann, 1996.

[5] T. Yao, S. Fukunaga, and T. Nakai, "Reliable broadcast message authentication in wireless sensor networks," in *Proc. Workshops on Emerging Directions in Embedded and Ubiquitous Computing*, 2006, pp. 271–280.

[6] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a tiny aggregation service for ad-hoc sensor networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 131–146, 2002.

[7] F. Bergadano, D. Cavagnino, and B. Crispo, "Chained stream authentication," in *Proc. Int. Workshop on Selected Areas in Cryptography*, 2001, pp. 144–157.

[8] T. Heer, S. Götz, O. Garcia-Morchon, and K. Wehrle, "Alpha: An adaptive and lightweight protocol for hop-by-hop authentication," in *Proc. Conf. on Emerging Networking Experiments and Technologies*, 2008.

[9] A. Perrig, R. Canetti, J. Tygar, and D. Song, "Efficient authentication and signing of multicast streams over lossy channels," in *Proc. IEEE Symp. on Security and Privacy*, 2000, pp. 56–73.

[10] D. Liu and P. Ning, "Multi-level uTESLA: Broadcast authentication for distributed sensor networks," *ACM Trans. in Embedded Computing Systems*, vol. 3, no. 4, pp. 800–836, 2004.

[11] Y. Chen, I. Lin, C. Lei, and Y. Liao, "Broadcast authentication in sensor networks using compressed bloom filters," in *Proc. Int. Conf. on Distributed Computing in Sensor Systems)*, 2008, pp. 99–111.

[12] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "LHAP: a lightweight hop-by-hop authentication protocol for ad-hoc networks," in *Proc. Int. Conf. on Distributed Computing Systems*, 2003, pp. 749–755.

[13] M. Luk, A. Perrig, and B. Whillock, "Seven cardinal properties of sensor network broadcast authentication," in *Proc. ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2006, pp. 147–156.

[14] Y. Desmedt, Y. Frankel, and M. Yung, "Multi-receiver/multi-sender network security: efficient authenticated multicast/feedback," in *Proc. Conf. of the IEEE Computer and Communications Societies*, 1992, pp. 2045–2054.

[15] R. Safavi-Naini and H. Wang, "New results on Multi-Receiver authentication codes," in *Proc. Int. Conf. on the Theory and Application of Cryptographic Techniques*, 1998, pp. 527–544.

[16] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas, "Multicast security: a taxonomy and some efficient constructions," in *Proc. Conf. of the IEEE Computer and Communications Societies*, 1999, pp. 708–716.

[17] W. Zhang, N. Subramanian, and G. Wang, "Lightweight and compromise-resilient message authentication in sensor networks," in *Proc. IEEE Conf. on Computer Communications*, 2008, pp. 1418–1426.

[18] D. Boneh, G. Durfee, and M. Franklin, "Lower bounds for multicast message authentication," in *Proc. Int. Conf. on the Theory and Application of Cryptographic Techniques*, 2001, pp. 437–452.

[19] R. Gennaro and P. Rohatgi, "How to sign digital streams," in *Proc. Int. Cryptology Conf.*, 1997, pp. 180–197.

[20] P. Golle and N. Modadugu, "Authenticating streamed data in the presence of random packet loss," in *Proc. Network and Distributed System Security Symp.*, 2001, pp. 13–22.

[21] J. M. Park, E. Chong, and H. Siegel, "Efficient multicast packet authentication using signature amortization," in *Proc. IEEE Symp. on Security and Privacy*, 2002, pp. 227–240.

[22] S. Miner and J. Staddon, "Graph-based authentication of digital streams," in *Proc. IEEE Symp. on Security and Privacy*, 2001, pp. 232–246.

[23] C. K. Wong and S. Lam, "Digital signatures for flows and multicasts," in *Proc. Int. Conf. on Network Protocols*, 1998, pp. 198–209.

[24] S. Zhu, S. Setia, S. Jajodia, and P. Ning, "An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor networks," in *Proc. IEEE Symp. on Security and Privacy*, 2004, pp. 259–271.

[25] C. Jutla, "PRF domain extension using dags," in *Proc. Theory of Cryptography Conf.*, 2006, pp. 561–580.

[26] H. Chan, "Authenticated communication and computation in known-topology networks with a trusted authority," PhD Dissertation, Carnegie Mellon University, Department of Computer Science, 2009.

[27] Zigbee Alliance, "Zigbee specification document 053474r17," 2008.

[28] H. Chan, A. Perrig, and D. Song, "Secure hierarchical in-network aggregation for sensor networks," in *Proc. ACM Conf. on Computer and Communications Security*, 2006, pp. 287–297.

[29] J. Katz and A. Lindell, "Aggregate message authentication codes," in *Proc. Cryptographers Track at the RSA Conf.*, 2008, pp. 155–169.