Secure Communication Over Radio Channels

Shlomi Dolev
Ben-Gurion University
Beer-Sheva, Israel
dolev@cs.bgu.ac.il

Seth Gilbert
EPFL IC
Lausanne, Switzerland
seth.gilbert@epfl.ch

Rachid Guerraoui
EPFL IC
Lausanne, Switzerland
rachid.guerraoui@epfl.ch

Calvin Newport
MIT CSAIL
Cambridge, MA, USA
cnewport@mit.edu

ABSTRACT

We study the problem of secure communication in a multi-channel, single-hop radio network with a malicious adversary that can cause collisions and spoof messages. We assume no pre-shared secrets or trusted-third-party infrastructure. The main contribution of this paper is f-AME: a randomized (f)ast-(A)uthenticated (M)essage (E)xchange protocol that enables nodes to exchange messages in a reliable and authenticated manner. It runs in $O(|E|t^2 \log n)$ time and has optimal resilience to disruption, where E is the set of pairs of nodes that need to swap messages, n is the total number of nodes, C the number of channels, and $t < C$ the number of channels on which the adversary can participate in each round. We show how to use f-AME to establish a shared secret group key, which can be used to implement a secure, reliable and authenticated long-lived communication service. The resulting service requires $O(nt^3 \log n)$ rounds for the setup phase, and $O(t \log n)$ rounds for an arbitrary pair to communicate. By contrast, existing solutions rely on pre-shared secrets, trusted third-party infrastructure, and/or the assumption that all interference is non-malicious.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems; C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*wireless communication*

General Terms

Algorithms, Security

Keywords

Wireless Radio Networks, Malicious (Byzantine) Interference, Randomized Algorithms

PODC'08, August 18–21, 2008, Toronto, Ontario, Canada.

1. INTRODUCTION

In recent years, wireless networking technology has held out the prospect of enabling communication without the need for physical network infrastructure. Protocols such as Bluetooth realize this promise by allowing ad hoc collections of nearby devices to easily construct a simple one-hop network (sometimes called a *piconet*). The radio medium, however, is publicly accessible. A committed malcontent can eavesdrop, interfere with broadcasts, and spoof messages. Shared secrets provide a standard mechanism for circumventing these types of attacks. If the honest devices share a secret key that is unknown to the adversary, they can eliminate eavesdropping and message spoofing by encrypting and signing their messages, respectively. Another common use of a shared secret key is to generate a pseudo-random channel-hopping pattern that allows the participants to efficiently avoid adversarial interference.

This approach, however, begs an obvious question: *How do we establish shared secrets?* Protocols such as Bluetooth, for example, require that a shared secret is manually entered into the relevant devices (in the form of a *passkey*). This might prove inconvenient or impossible in many situations. Furthermore, even when pre-programming is feasible, it might be useful to be able to re-key dynamically, for example, after the detection of a compromised device. Another approach is to rely on trusted third-party infrastructure, such as a PKI, to authenticate and secure communication. In many settings, however, such services are unavailable.

Authenticated Message Exchange

In this paper we address this crucial question. We begin by studying the problem of Authenticated Message Exchange (AME). Initially, we are given an arbitrary set E of pairs of devices that want to exchange information. The goal of AME is for as many of these pairs as is possible to communicate messages in an authenticated manner. For the purpose of this paper, we assume that there are $t + 1$ channels available for communication and that the adversary can affect up to t channels per round, causing interference or spoofing messages. (Notice that this is the minimal number of channels for which any communication is feasible.)

A significant challenge lies in efficiently achieving authenticated communication. A deterministic algorithm can read-

ily achieve authentication by relying on a pre-determined broadcast schedule: If p_i is known to be broadcasting on a given channel in a given round, the worst the adversary can do is broadcast concurrently and cause a collision; if another node p_j receives a message on that channel in that round, it can conclude that the message was sent by p_i itself, not the adversary. A purely randomized approach, on the other hand, is difficult to authenticate since the receiver cannot determine whether the adversasry or the honest node was broadcasting on that channel: with some probability the honest node sent the message, but with some probability the honest node was on a diffrent channel, allowing the adversary to spoof messages. Unfortunately, we conjecture that a purely deterministic solution would require exponential time to solve AME. In this paper we identify a middle-ground: harnessing the efficiency of randomization for transmitting data, while leveraging deterministic broadcast scheduling to authenticate the messages.

We describe a randomized protocol called f-AME (fast-Authenticated Message Exchange) that solves the AME problem with optimal resilience: all but at most t devices successfully communicate as specified by the set E. The f-AME protocol requires no pre-shared secrets or trusted third-party infrastructure. It terminates in $O(|E|t^2 \log n)$ rounds. By contrast, relying on an all-to-all gossip protocol, as can be found in [13], would result an exponential running time, and yet still fail to achieve optimal resilience.

The core insights behind f-AME are graph theoretic. To best capture this connection, we first define an abstract graph-theoretic game that we call *Starred-Edge Removal*. We then describe a greedy strategy that efficiently solves the game. Our final f-AME protocol simulates the game on the multi-channel network, using a randomized feedback routine to synchronize the distributed view of the game and maintain a correspondence between the game state and the AME message set. We show that our efficient strategy for the edge-removal game induces an optimally resilient solution to the AME problem.

Established a Secret Group Key

We then use f-AME to establish a shared secret group key. The algorithm begins with a setup phase in which secure pairwise keys are established. We initialize f-AME with messages generated by a one-round cryptographic key-exchange protocol (such as Diffie-Hellman) and a set E of pairs describing a sparse $t+1$-connected graph with $n(t+1)$ edges that we call a "$(t+1)$-leader spanner." The resulting pairwise shared keys are used to distribute up to $t+1$ proposals for the shared group key. An agreement protocol allows the nodes to safely agree on a single proposal to adopt. The total running time to establish the shared group key is $O(nt^3 \log n)$ rounds.

Long-Lived Communication Service

Shared secret group keys have a variety of uses. In this paper we describe one: the construction of a secure, reliable, and authenticated long-lived communication service. Once a group key is established, the service requires only $\Theta(t \log n)$ rounds for an arbitrary pair to communicate. It differs from f-AME in that it is long-lived, rather than single-shot. Moreover, after the setup phase (which relies on f-AME to establish a group key), the nodes can exchange messages more efficiently than re-running the f-AME protocol. Also unlike f-AME, any pair can communicate whenever it chooses—even if no other nodes are interested in communicating at that time. (By contrast, f-AME relies on the fact that many pairs want to communicate, thus preventing the adversary from blocking all of them.) This represents, to the best of our knowledge, the first such communication primitive for a radio network with malicious interference and no *a priori* shared secrets or trusted infrastructure.

2. RELATED WORK

There exists much research in broadcast and gossip in the context of single-channel radio networks (e.g., [1–3, 7, 8, 11, 15, 18, 20, 23]). Much of this research focuses on the problem of channel contention, and assumes reliable devices in a non-malicious environment. Recently, there has been some interest in crash-tolerance in radio networks (e.g., [9,10,19]), and Byzantine-resilient broadcast in radio networks [4, 16]. In these latter studies, however, the adversary cannot disrupt communication. When adversarial disruption is allowed, there exist three common approaches in the literature. The first assumes that messages may be corrupted *at random* (e.g., [21]). The second bounds the number of messages that the adversary can transmit or disrupt, due, for example, to a limited energy budget (e.g., [14, 17]). The third assumes some manner of shared secrets. There have been proposed a variety of approaches for establishing such a shared secret, all of which require some form of human intervention (e.g., the manually configured *passkeys* of Bluetooth [5]), or out-of-band communication (e.g., physical contact between devices [22], a limited-bandwidth secure side-channel [6, 12], location information [6] and strong collision detection [6, 14]).

The present paper, along with [13], are the first, to our knowledge, to consider multi-channel networks subject to malicious disruption in which processes do not possess *a priori* shared secrets (or have access to out-of-band means for establishing these secrets). Dolev et al. [13] consider the problem of *almost gossip*, in which all but t rumors are delivered to all but t nodes. They focus on oblivious algorithms that do not adapt to the execution in progress, and show for the special case of $t = 1$ that there exists a tight bound for gossip of $\Theta(n^2/C^2)$. They extend their algorithm for general t, achieving running time $O((en/t)^{t+1})$. Thus using gossip to solve AME would be quite slow, and also would only achieve suboptimal resilience, i.e., $2t$-disruptability (see Section 4 for more details).

3. MODEL

We assume a single-hop radio network with n devices (which we call "nodes"), described by $\Pi = \{p_1, ..., p_n\}$. The network has $C > 1$ communication channels. Executions proceed in synchronous rounds; all nodes start in the same round. During each round, each node can choose to transmit or receive on a single channel. If a single node transmits on a channel, all receivers on that channel receive the transmission. If no node, or two or more nodes transmit on the same channel, then the receivers on that channel receive nothing. (We do not assume that nodes can detect collisions.)

We also assume the presence of an adversary that can transmit on up to $t < C$ channels during each round, and can listen on all C channels. The adversary can therefore disrupt communication in two ways: (1) *jamming*: by transmitting

concurrently with an honest node, the adversary causes a message to be lost due to collision; and (2) *spoofing*: by transmitting a fake message on an otherwise empty channel, the adversary causes a node to receive incorrect information. Notice that since communication is not authenticated, a node cannot definitively determine who sent a given message. As is typical, we assume the adversary does not know in advance the honest nodes' random choices; only by observing a node's actions is it possible to determine which random choice it made. Without loss of generality, we assume that at the end of each round, the adversary learns all random choices made in all completed rounds.

4. PRELIMINARIES

In this paper, we focus on the case where $C = t + 1$; notice that this is the minimum number of channels for which any communication is possible. Where relevant, we briefly discuss how our results can be improved for larger values of C, specifically, the case where $C = 2t$. We also fix n to be sufficiently large with respect to t, specifically $n > 3(t+1)^2 + 2(t+1)$. (This is not unreasonable, since n is typically large with respect to the number of channels C, and $t < C$.)

The focus of this paper is a problem we call *Authenticated Message Exchange (AME)*. An AME protocol is initially provided with a set E of ordered pairs. For each ordered pair $(v, w) \in E$ we attempt to send a message $m_{v,w}$ from v to w. The AME protocol guarantees three properties: (1) *authenticity*, i.e., a node receives only authentic messages and successfully ignores messages spoofed by the adversary; (2) *sender awareness*, i.e., each node knows which of their messages were successfully received; and (3) *limited disruptability*, i.e., the adversary can disrupt only a limited number of the nodes that want to communicate. More formally, an AME protocol makes the following guarantees:

Definition 1 (d-Disruptable AME Protocol). *For d a non-negative integer, we say that protocol $\mathcal{P}$ is a **d-disruptable AME protocol** if for every set E of ordered pairs of distinct nodes from Π, where $|E| \geq d$, and for every $(v, w) \in E$, message $m_{v,w}$ is initially known only to v, the following properties hold with high probability:*

1. **Authentication:** *For each $(v, w) \in E$: w outputs either $\langle (v, w), \mathsf{fail} \rangle$ or $\langle (v, w), m_{v,w} \rangle$.*
2. **Sender Awareness:** *For each $(v, w) \in E$: v can determine whether j output $\langle (v, w), \mathsf{fail} \rangle$ or $\langle (v, w), m_{v,w} \rangle$.*
3. **d-Disruptability:** *Let $E' \subseteq E$ be the subset of pairs that output* fail. *The minimum vertex cover of the* disruption graph *$G_d = (\Pi, E')$ contains no more than d vertices.*

Notice that the disruptability property describes the number of "failures" that can be induced by the adversary. That is, for a d-disruptable protocol, there exists a set of no more than d nodes such that if we remove these nodes from consideration, all other nodes succeed in communicating their messages. We observe that no protocol can achieve better than t-disruptability; this implies that the adversary can always prevent at least t of the nodes from communicating successfully. Notice that in the case of deterministic AME protocols, this is immediately clear since the adversary can readily target t of the nodes; slightly more care is needed in the case of randomized protocols, but the same argument holds.

Theorem 2. *For $n \geq 2t$ and $d < t$: there exists no d-disruptable Authenticated Message Exchange protocol.*

Proof. Assume for the sake of contradiction that such a protocol $\mathcal{P}$ exists. Fix $E = \{(i, i + t) | i \in \{1, \ldots, t\}\}$, that is, there are t disjoint pairs of nodes that want to communicate. Consider an execution α_1 of $\mathcal{P}$ in which the adversary simulates nodes 1 through t and attempts to disseminate a fake message (different from the real message, in each case); whenever a node $i \in \{1, \ldots, t\}$ broadcasts a message on a deterministically chosen channel, the adversary does the same; whenever i broadcasts a message on a randomly chosen channel, the adversary also chooses a channel at random based on the same distribution. Because the adversary can broadcast on up to t different channels per round, it can carry out this simulation faithfully.

Property (3) of the AME definition requires that at least one pair in E does not output fail, since $|E| = t$ and $d < t$. Without loss of generality, assume this is pair $(1, t + 1)$. Property (1) requires that, with high probability, $t + 1$ outputs $m_{1,t+1}$. Let r_1 be the string of random bits used by process 1, and let r_A be the string of random bits used by the adversary. To process $t + 1$, however, this execution α_1 is indistinguishable from another execution α_2 in which the adversary chooses random bits r_1 and process 1 chooses random bits r_A. Moreover, both executions α_1 and α_2 occur with equal probability. It follows, therefore, that node $t + 1$ has equal probability of outputting the correct message, $m_{1,t+1}$, and the fake message generated by the adversary. This violates property (1), implying a contradiction. □

5. AUTHENTICATED MESSAGE EXCHANGE

In this section, we develop an Authenticated Message Exchange (AME) protocol with optimal resilience, i.e., the protocol is at most t-disruptable. Throughout this section, fix $G^{AME} = (\Pi, E^{AME})$ to be the directed graph representing the pairs that have messages to exchange, and let $n = |\Pi|$. For every $(v, w) \in E^{AME}$, let $m_{v,w}$ be the message that v is sending to w. We call our protocol f-AME (fast-Authenticated Message Exchange) as it runs in time $O(|E^{AME}|t^2 \log n)$ rounds. Our f-AME protocol is built on two insights.

The first insight is related to authentication: a good method for achieving authenticated communication is to schedule $t + 1$ sender/receiver pairs to communicate concurrently; each sender then transmits its message directly to the receiver. The adversary cannot simultaneously block all $t + 1$ channels, and hence at least one message is successfully communicated. If the channel assignment is determined deterministically and in advance, then the adversary can disrupt communication but not spoof messages, as any broadcast by the adversary will simply cause a collision.

This strategy alone, in which each message is transmitted directly from the source to the destination, can achieve no better than $2t$-disruptability. Consider a complete communication graph. The adversary can identify t disjoint sets of three processes; it can block any channel which contains two or more processes from the same set of three. The result: a disruption graph with t edge-disjoint triangles, inducing minimum vertex cover of size $2t$.

We turn, therefore, to our second insight: to further reduce disruptability further, a node can recruit *surrogates* to relay messages on its behalf. These surrogates later allow us to schedule two edges to broadcast simultaneously, even if they share a source. (Notice that by routing messages via surrogates, the adversarial strategy of isolating triangles, as described above, is no longer feasible.)

Roadmap

To best clarify our deployment of these insights, we present our protocol in stages. First, in Section 5.1, we define an abstract graph-theoretic "game" that we call *starred-edge removal*. This game isolates the core scheduling concerns of removing edges from the disruption graph and recruiting surrogates from the more practical concerns of coordinating distributed nodes on a multi-channel network. In Section 5.2, we describe an efficient strategy, greedy-removal, that solves the starred-edge removal game in a minimal number of rounds. In the final two sections, Sections 5.3 and 5.4, we describe a distributed simulation of the starred-edge removal game that maintains a correspondence between the game graph and the AME disruption graph. At the core of the distributed simulation is a randomized sub-routine, communication-feedback, that informs all nodes, with high probability, which channels were disrupted during a given round.

5.1 The (G, t)-starred-edge removal game

In this section, we define the (G, t)-starred-edge removal game, where $G = (V, E)$ is a directed graph and t is a natural number less than $n = |V|$. As a point of notation, for every edge $(v, w) \in E$, we refer to v as the *source* and w as the *destination*. Throughout the game, we also maintain an auxiliary set $S \subseteq V$, initially empty. (If a node $v \in V$ is in the set S, we say that v is *starred*.) The game consists of a series of rounds. In each round:

1. The *player* proposes a set P that is comprised of the following: (1) nodes from V; and (2) edges from E. The proposal must satisfy the *proposal restrictions* described below.

2. The *referee* responds by choosing a non-empty subset of nodes and edges from P.

3. For every node chosen by the referee, the player adds the node to S. For every edge chosen by the referee, the edge is removed from E.

The game terminates if the player succeeds in removing enough edges from G to produce a new graph with a vertex cover of size at most t. The goal of the referee, by contrast, is to delay the player from completing the game for as long as possible.

Proposal Restrictions

It remains to specify the restrictions on the player's proposed set P of nodes and edges. They are as follows:

1. The set P must consist of exactly $t + 1$ items, each of which is either a node in V or an edge in E.

2. Every node in P is unique, i.e., it does not appear as either the source or destination of any edge in P.

3. No two edges in P share a destination.

4. Two edges in P share a source $v \in V$ only if $v \in S$.

For example, if v, w, z are nodes in V, Restriction 3 prohibits the player from proposing both edges (v, z) and (w, z). Similarly, Restriction 4 prohibits the player from proposing both edges (v, w) and (v, z) unless node $v \in S$. Roughly speaking, a node being added to S corresponds to the fact that it has successfully recruited surrogates to broadcast on its behalf (which is required to achieve t-resilience, as per our discussion above).

5.2 The Greedy Removal Strategy

In this section, we describe a centralized strategy, which we call greedy-removal, that solves the (G, t)-starred-edge removal game in $O(|E|)$ rounds. Our strategy ensures that in each round of the game, either a new edge is removed from E or a new node is added to S. The greedy-removal strategy works as follows. Define two sets P_1 and P_2, with respect to graph $G = (V, E)$:

- $P_1 = \{v \in V \backslash S : (v, *) \in E\}$. That is, P_1 consists of the set of nodes not in S that are the source of some edge in E.

- $P_2 = \{(v, w) \in E : v, w \notin P_1\}$. That is, P_2 consists of edges in which neither source nor destination is in P_1. This implies that if $(v, w) \in P_2$, then $v \in S$.

The player chooses any arbitrary subset of $t + 1$ elements from $P_1 \cup P_2$ that satisfy Proposal Restrictions 1–4 of the (G, t)-starred-edge removal game. If there is no subset of $P_1 \cup P_2$ of size at least $t + 1$ that satisfies Restrictions 1–4, we say that the greedy-removal strategy has *terminated*. The key result in this section is that the greedy-removal strategy terminates only if the minimum vertex cover of G is no greater than t; this implies that the game is complete.

LEMMA 3. *Assume that no subset $P \subseteq P_1 \cup P_2$ satisfies Restrictions 1–4 of the (G, t)-starred-edge removal game. Then graph G has a minimum vertex cover no larger than t.*

PROOF. We proceed in the following three steps: (1) constructing a set $V' \subseteq V$; (2) arguing that it is of size at most t; and (3) showing that it is a vertex cover for G. We define the set of vertices V' as follows:

$$V' = P_1 \cup \{w : (*, w) \in P_2\}$$

Every node in P_1 is included in the set V'; in addition every destination for every edge in P_2 is also included in V'. We first argue that the number of destinations in P_2 is at most $t - |P_1|$; we can derive from this that $|V'| \leq t$. Assume for the sake of contradiction that there exist at least $t - |P_1| + 1$ destinations in P_2. We can define the following proposal P of size $t + 1$: all the nodes in P_1, along with the $t - |P_1| + 1$ destination-disjoint edges from P_2. It follows immediately that this set satisfies Restriction 1. Restriction 2 follows from the fact that every edge in P_2 is disjoint from every node in P_1. Restriction 3 follows from the fact that we have chosen at most one edge in P_2 for each destination. Restriction 4 is satisfied by the fact the source of every edge in P_2 is also in S. (Otherwise, the source of the edge would be included in P_1). The existence of this proposal P of size $t+1$ contradicts our assumption that the player has no legal

Figure 1: Detect collisions.

```
1  communication-feedback(W, b)_i
2      D ←∅
3       // send feedback for each channel
4      for r =1 to C
5          repeat Θ(C/(C−t) lg n):
6              // if p_i is a witness for r
7              if p_i ∈W[r] then
8                  // if b = false send "false"
9                  if b = false then
10                     let k =rank(p_i, W[r])
11                     bcast(⟨false⟩, k);
12                 // if b = true send "true"
13                 else
14                     D ←D ∪{r}
15                     let k =rank(p_i, W[r])
16                     bcast(⟨true, r⟩, k)
17             // if p_i is not a witness for r
18             else
19                 Choose channel k at random from [1,C]
20                 reports ←reports ∪ recv(k)
21         if ⟨true, r⟩ ∈reports then
22             D ←D ∪{r}
23     return D
```

move available, from which we conclude that V' is of size at most t.

Finally, we argue that V' is a vertex cover for G. In particular, consider some edge $(v, w) \in E$. If $(v, w) \in P_2$, then, by definition, $w \in V'$. Therefore, (v, w) is covered by V'. Assume, then, that $(v, w) \notin P_2$. By the definition of P_2: either v or w must therefore be in P_1. This implies however that either v or w is in V', once again covering the edge. Thus, we have demonstrated a vertex cover of size at most t. □

THEOREM 4. *The* greedy-removal *strategy solves the* (G, t)*-starred-edge removal game in* $O(|E|)$ *moves.*

PROOF. If the referee returns an edge from P_2, this removes an edge from E. If the referee returns a node from P_1, this adds a new node to S (as every node in P_1 is not in S). Since at most $|E|$ edges can be removed from E, and at most $2|E|$ nodes can be added to S, the desired bound holds. □

5.3 Communication Feedback

Our f-AME protocol simulates the starred-edge removal game. A key sub-routine needed by this simulation is a method for nodes to agree on which channels were disrupted during a given round. The communication-feedback protocol satisfies this need.

Specifically, after executing a round of communication, the nodes call the feedback routine communication-feedback in order to agree on which channels have been disrupted; the feedback routine takes two parameters: (1) W, a partition of the processes $\{p_1, \ldots, p_{C^2}\}$ into C sets of size C; (2) b, a binary flag, either true or false. Intuitively, W assigns as set of C "witnesses" to each of the C channels, and the flag b indicates whether the caller of the feedback routine has received a message in that round of communication.

We assume that all processes call communication-feedback in the same round with the same partition W. For each $c \in \{1, \ldots, C\}$, we refer to the processes $W[c]$ as *witnesses* for channel c. We also assume that every witness for channel c has the same value of the flag b, which we refer to as the *channel* c *flag*. For every $c \in \{1, \ldots, C\}$, let b_c describe the channel c flag. We show that under these circumstances, the call to communication-feedback satisfies the following:

LEMMA 5. *Each invocation of* communication-feedback *terminates in time* $O(t^2 \log n)$*, and returns a set* D *satisfying the following property, with high probability: for every channel* $c \in \{1, \ldots, C\}$*,* $c \in D$ *if and only if* $b_c =$ true.

PROOF. Fix come channel $c \in \{1, \ldots, C\}$, and some node $p_j \in \Pi$ that invokes communication-feedback. It is clear by inspection that the feedback routine terminates in $C \cdot \frac{C \log n}{C-t}$ rounds, which is $O(t^2 \log n)$. We now examine the set D produced when p_j calls communication-feedback.

First, assume that $b_c =$ false. Consider the iteration of lines 4–22 where $r = c$, as this is the only instance in which c can be added to the set D. Notice that in each of the $\Theta(t \log n)$ rounds that take place where $r = c$, for each channel $k \in \{1, \ldots, C\}$, one of the witnesses in $W[c]$ broadcasts a false message on channel k (line 11). Thus it is impossible for p_j to receive a $\langle \mathsf{true}, c \rangle$ message, and hence $c \notin D$ for node p_j. (Notice, the adversary can broadcast $\langle \mathsf{true}, c \rangle$, but because every channel is occupied by a broadcasting witness, this will lead only to a collision.)

Consider instead the case in which the value $b_c =$ true. If p_j is a witness for c, then p_j, immediately adds r to D (line 14). Otherwise, we will argue that, with high probability, p_j receives a $\langle \mathsf{true}, c \rangle$ message from some witness for channel c. Specifically, for each of the $\Theta(\frac{C}{C-t} \log n)$ rounds that take place where $r = c$, one witness on each channel is broadcasting $\langle \mathsf{true}, c \rangle$. The process p_j chooses a random channel on which to listen (line 20). As a result, we note that in each round, p_j has probability no worse than $(C - t)/C$ of selecting a channel not disrupted by the adversary and therefore receiving a $\langle \mathsf{true}, c \rangle$ message. Receiving this message causes p_j to add c to the set D.

It follows that over $\Theta(\frac{C}{C-t} \log n)$ rounds, we can conclude, by a straightforward Chernoff bound that p_j receives $\langle \mathsf{true}, c \rangle$, and thus adds c to D, with high probability. By a union bound over all nodes, we conclude that every $p_j \in \Pi$ that is not a witness receives such a message with high probability. □

5.4 The fast-AME Protocol

Here were bring together all the pieces to present f-AME (fast-Authenticated Message Exchange). The protocol is t-disruptable and completes in $O(|E|t^2 \log n)$ rounds. At the core of our protocol is a distributed simulation of the (G, t)-starred-edge removal game, where, initially, $G = G^{AME}$. The simulation maintains a *graph equivalence* invariant: after r simulated rounds of the game, the graph G in the game is equivalent to the current disruption graph. Since the starred-edge removal game terminates only when the vertex cover is no greater than t, the game simulation results in t-disruptability.

Overview:

The f-AME protocol, described at a high level in Figure 2, proceeds by simulating moves in the starred-edge removal game. Each move is simulated using $\Theta(t^2 \log n)$ rounds of communication, and consists of two phases: a message-

Figure 2: Structure for f-AME Protocol.

$SIM \leftarrow$ new instance of (G^{AME}, t)-starred edge removal.
while (SIM not terminated):
1. Apply greedy-removal to SIM to determine the proposal P.
2. Broadcast according to P.
3. Call collision-feedback to determine referee response R.
4. Update SIM according to R.

transmission phase (which requires one round of communication), and a feedback phase (which requires $\Theta(t^2 \log n)$ rounds of communication). In the first phase, the nodes simulate the player's proposal in the game, choosing a set P and transmitting messages accordingly. In the second phase, the nodes simulate the referee, using the communication-feedback routine to agree on the set returned by the referee.

State:

Each node $p_j \in \Pi$ maintains the following state throughout the execution of the protocol: a graph $G_j = (V_j, E_j)$; and a set S_j. These represent p_j's local copy of the starred-edge removal game. Initially, for every p_j, $G_j = G^{AME}$ and $S_j = \emptyset$.

Message-transmission phase:

The message transmission phase proceeds as follows. Each node p_j locally applies the greedy-removal strategy to its local copy of G_j and S_j to determine a valid proposal P_j. We will argue in our correctness proof that every node generates the same proposal P_j, and hence all the nodes behave consistently in the rounds that follows. We now describe how a node calculates which nodes should broadcast or listen during the message-transmission phase, and which channels these nodes should use. Notice that given P_j, node p_j can determine the precise behavior of every other node, and thus can determine the appropriate action to take during the message-transmission phase.

Recall that P_j consists of both nodes and edges. Each node in P_j is scheduled for one of the $t+1$ channels on which to broadcast during the message-transmission phase. Ideally, we would also schedule each edge in P_j on one of the $t+1$ channels: for each edge, the source would choose a channel on which to broadcast, and the destination would listen on the same channel.

It may, however, be the case that some of the edges in P_j share the same source, and hence cannot be scheduled simultaneously. We know, however, by the restrictions of the starred-edge removal game that none of the edges in P_j share the same destination; nor do any of the edges in P_j contain a node in P_j.

Consider the case where P_j includes two edges that share a source, say (v, w) and (v, z). In this case, we know that $v \in S$, by the restrictions of the game. We will show that every node in S has at least $3(t+1)$ surrogates. Since there are at most $2(t+1)$ nodes involved with P_j as either a node, a source, or a destination, we can conclude that there are at least $t+1$ nodes available to act as surrogates for v. Thus, for every edge in P_j, either the source or one of its available surrogates is scheduled to broadcast on a channel, and we schedule the destination to receive on the same channel.

Lastly, $3(t+1)$ witnesses are scheduled to listen on each of the $t+1$ channels that are being used. Since $n > 3(t+1)^2 + 2(t+1)$, and since at most $2(t+1)$ nodes have, to this point, been assigned either to broadcast or to listen, it is clear that there are enough nodes to satisfy this assignment. Of these witnesses, for each channel $c \in \{1, \ldots, C\}$, we choose a subset of $t+1$ nodes per channel and assign these to $W[c]$ for use in the call to communication-feedback that occurs during the next phase.

To this point, we have described the choice of nodes to broadcast and listen in a centralized fashion as a function of P_j. Node p_j behaves according to the schedule which it calculates locally from P_j. As we show that every node calculates the same P_j, we can conclude that every node calculates the same scheduling of channels, thus leading to consistent behavior that avoids contention.

Thus, to summarize a node's behavior in the message-transmission phase: each node p_j examines the schedule that has been determined by P_j and acts accordingly. If p_j is scheduled to broadcast its own value on some channel c, then it transmits the vector of all values $m_{j,*}$ on channel c. If it is scheduled to broadcast as a surrogate for p_w on some channel c, then p_j transmits the vector of all the values $m_{w,*}$. If it is scheduled to receive on some channel c, then it does so. (In Section 5.6 we reduce the message size to constant.) When the round completes, at least one of the $t+1$ channels has transmitted successfully; the remaining t may have been disrupted by the adversary. Notice that since $t+1$ honest nodes broadcast on the $t+1$ channels, then the adversary cannot successfully spoof messages, as every one of the $t+1$ channels on which nodes receive is in use.

Feedback phase:

In the second phase of simulating the move in the starred-edge removal game, the nodes run communication-feedback, using the witness set W as defined above. Each witness sets its flag as follows: if it received a message during the message transmission phase, then it sets its boolean flag to true; otherwise, it sets its flag to false. Notice that every witness for a given channel c chooses the same setting for its flag. The communication-feedback routine returns to each process p_j some set D_j. We will argue that, as per the guarantees of communication-feedback, every set D_j is identical.

Each node p_j then simulates the referee returning the following set of nodes and edges from P_j: (1) every node $v \in P_j$ where v was scheduled to broadcast during the message-transmission round on channel c and $c \in D_j$; (2) every edge $(w, z) \in P_j$ where z was scheduled to receive on channel c and $c \in D_j$.[1]

THEOREM 6. *f-AME is a t-disruptable Authenticated Message Exchange protocol that terminates in $O\left(|E|t^2 \log n\right)$ rounds of communication.*

PROOF. We show that throughout the starred-edge removal game round simulation, the following three invariants are maintained:

[1]Notice that we could simplify the algorithm somewhat by having communication-feedback return the actual messages received on each non-disrupted channel, rather than simply the single bit of information indicating success or failure. The running time, however, would remain unchanged. We use the binary version of the communication-feedback as it seems from preliminary research to be more robust to harsher adversary models, including, for example, Byzantine node corruptions.

(Invariant 1.) At the beginning of each simulated move of the starred-edge removal game, every node has the same simulated game graph G , and the same starred set S. This implies that the same nodes and edges have been removed in the game up until this round, and the same nodes have been added to S.

(Invariant 2.) For every node $p_j \in \Pi$, for every node $v \in S_j$ (the starred-node set of node p_j), and for every w such that $(v, w) \in E$: the message $m_{v,w}$ is known to at least $3(t + 1)$ nodes.

(Invariant 3.) After every simulated move of the starred-edge removal game, for every process p_j, the graph G_j represents the disruption graph for the AME problem. That is, if $(v, w) \in E^{AME}$ and w has not yet learned $m_{v,w}$, then edge (v, w) is in G_j.

It is easy to see that all three invariants are initially true: Invariant 1 follows since G_j and S_j are initialized identically for every node p_j; Invariant 2 follows since S_j is initially empty, for every node p_j; Invariant 3 follows since initially $G_j = G^{AME}$ for every node p_j.

We proceed inductively to show that these invariants are maintained, with high probability, after each simulated move of the starred-edge removal game. We first note that since, at the beginning of the simulated round, every node p_j has the same graph G_j and the same set S_j, by Invariant 1, every node p_j chooses the same set P_j when executing the greedy-removal strategy. As a result, all the nodes calculate the same broadcast/receive schedule for the message-transmission phase, and hence exactly one node broadcasts on each of the $t + 1$ channels. This also implies that every node calculates the same witness set $W[c]$ for channel $c \in \{1, \ldots, C\}$.

We therefore conclude that each of the nodes scheduled to receive on some channel either receives a message from an honest transmitter, or receives nothing—in the case of an adversarial disruption. (Thus, adversarial spoofing is impossible here.) Since the adversary can disrupt at most t channels, at least one set of nodes scheduled to listen does in fact receive a message in the message-transmission phase.

Moreover, we observe that every witness for a given channel $c \in \{1, \ldots, C\}$ receives the same message-or-silence during the message-transmission phase, and hence initiates the communicate-feedback routine with the same setting for its flag. Thus, we conclude by Lemma 5 that, with high probability, every node p_j outputs $c \in D_j$ if and only if the broadcast on channel c succeeded during the message transmission round. As a result, every node outputs the same set D_j, and $D_j \neq \emptyset$.

Since each node p_j then simulates the referee in the same manner based on the same sets D_j and P_j, we can conclude that each node p_j updates its graph G_j and its set S_j in the same manner, maintaining Invariant 1.

Next, we argue that the second invariant is maintained: a node v in the graph G_j is added by node p_j to set S_j only if it was scheduled during the message transmission round to broadcast on some channel c, and only if $c \in D_j$ after the communication-feedback routine. This latter fact implies that the broadcast succeeded (as per Lemma 5), and hence that at least $3(t+1)$ witnesses received the message from v.

Finally, we note that the third invariant follows from the fact that a node p_j removes edge (v, w) from G_j only if the channel c on which v—or its surrogate—transmits $m_{v,w}$ to w is not disrupted, i.e., if $c \in D_j$.

Having shown that each of the three invariants is maintained throughout the simulation, it remains to observe that this implies the correctness of the algorithm, and to calculate the running time.

First, note that for each of the $O(|E|)$ game rounds required by the greedy-removal strategy, the simulation succeeds with high probability. We conclude by a union bound over the $O(|E|)$ rounds that each round is simulated correctly with high probability.

Second, we notice that by Theorem 4, at the end of $|E|$ simulated rounds, for every process p_j, the vertex cover for the graph G_j is at most t. Invariant 3 implies an equivalence with the AME disruption graph, leading to the conclusion that f-AME satisfies t-disruptability. Finally, each simulated game round requires $O(t^2 \log n)$ rounds of communication: communication-feedback requires $O(t \log n)$ rounds to deliver the feedback for each of $t+1$ channels, as described in Lemma 5. We conclude that the simulation of the $O(|E|)$ game rounds requires $O(|E|t^2 \log n)$ rounds. □

5.5 Optimizing f-AME for More Channels

For our analysis of f-AME, we assumed that $C = t + 1$, the maximum value of t for which any communication is possible. We have focused on this case in order to highlight that polynomial-time solutions exist even under worst-case interference. It is natural, however, to ask how performance might improve if more channels are available. Here we briefly consider two additional cases (the results are summarized in Figure 5.5).

Case 1: $C \geq 2t$.

Assume that the protocol uses exactly $C = 2t$ of the available channels. In this case, each proposal in the starred-edge removal game consists of $2t$ elements, and the referee returns t elements from each proposal. Thus, the greedy strategy terminates in $O(|E|/t)$ rounds. We next note that in each round of the communication-feedback routine, a listening node (that is not a witness) has a probability $\geq 1/2$ of avoiding disruption, and thus receives the feedback in $O(\log n)$ rounds with high probability. Thus, the total running time for acquiring feedback on all $2t$ channels is $O(t \log n)$. As a result, the overall running time of the protocol is $O(|E| \log n)$.

Case 2: $C \geq 2t^2$.

Fix $C' = \lfloor C/t \rfloor$. In the message-transmission phase we use only C' of the available channels; we refer to these as the *proposal channels*. In the feedback phase, we use all C channels. Thus, in the simulated starred-edge removal game, each proposal consists of C' elements; the referee responds with $C' - t$ elements of the proposal. Thus, the greedy strategy now terminates in $O(|E|/(C' - t)) = O(|E|/t)$ rounds.

The key, then, to obtaining better performance is to reduce the cost of feedback to $O(\log^2 n)$ rounds. Recall that for each channel, feedback can be accomplished in $O(\log n)$ rounds. Instead of executing the $O(C')$ instances of feedback sequentially (which would require $O(t \log n)$ time), we execute the feedback instances in parallel, using a parallel-prefix tree to merge information until all the information is known to a single set of witness; these witness can then disseminate the feedback to everyone else.

Number of channels	greedy-removal	communication-feedback (per invocation)	f-AME
$C \geq t+1$	$O(\lvert E\rvert)$	$O\left(t^2 \log n\right)$	$O\left(\lvert E\rvert t^2 \log n\right)$
$C \geq 2t$	$O\left(\frac{\lvert E\rvert}{t}\right)$	$O(t \log n)$	$O(\lvert E\rvert \log n)$
$C \geq 2t^2$	$O\left(\frac{\lvert E\rvert}{t}\right)$	$O\left(\log^2 n\right)$	$O\left(\frac{\lvert E\rvert \log^2 n}{t}\right)$

Figure 3: Optimizing f-AME time complexity for larger number of channels.

In more detail, begin by pairing up the proposal channels, and assign each pair (c_1, c_2) a unique set of t channels. Using only the assigned channels, the witnesses for proposal channel c_1 use the communication-feedback routine to propagate information to the witnesses for proposal channel c_2. Next, conversely, witnesses for proposal channel c_2 use the communication-feedback routine to propagate information to the witnesses for proposal channel c_1. At this point, the witnesses are informed of the feedback for both channels c_1 and c_2. Each pair $\langle c_1, c_2\rangle$ is now assigned a companion pair $\langle c_3, c_4\rangle$, and the process continues merging information until there are $t+1$ witnesses that have all the feedback. A final instance of communication-feedback is used for these witnesses to propagate the feedback to everyone else. The total time for the modified feedback routine is now $O(\log n \log C') = O(\log^2 n)$. The running time for the entire game simulation is now $O(\frac{|E|}{t} \log^2 n)$.

A Note on Optimality

Notice that a trivial lower for authenticated message exchange is $\Omega\left(\frac{|E|}{C}\right)$. In particular, consider a graph E consisting only of disjoint edges; at most C of the sources can transmit information in each round. When $C \geq 2t$, the optimized f-AME algorithm is within a factor of $t \log n$ of achieving this lower bound; when $C \geq 2t^2$, it is within a factor of $t \log^2 n$. Closing this gap, perhaps by deriving a non-trivial lower bound, remains an open question.

5.6 Optimizing f-AME Message Size

Each message of the f-AME protocol may be quite large. In fact, each message sent by process p_v may include up to $n-1$ messages of the form $m_{v,*}$. Here we describe an optimization that reduces the size of a protocol message to include no more than a constant number of AME values.

The optimized protocol is divided into two parts. In the first part, the *message gossip* phase, nodes exchange messages using a randomized gossip protocol. This ensures that every node has received every message; however it does not guarantee any authentication. In the second part, the original f-AME protocol is used to exchange shorter digests that can be use to authenticate the true messages from the gossip phase. In order to accomplish this, we will rely on secure hash functions. Let H_1 and H_2 be two such functions. Fix an edge set E, and let $E_v \subseteq E$ be the edges of the form $(v, *)$. Let M_v be an ordered sequence of the values to be sent on the edges in E_v.

Message Gossip Phase

Each edge $(v, w) \in E$ is given one epoch of length $\Theta(t^2 \log n)$ during which v communicates $m_{v,w}$. To reduce the number of fake messages assumed to be potentially true, nodes append some extra digest information onto each of these transmissions. Specifically, fix some node $v \in \Pi$, and let $\{m_{v,1}, \ldots, m_{v,k}\} = M_v$. Assume M_v describes the order of the epochs for these messages, that is, the epoch for $(v, 1)$ precedes the epoch for $(v, 2)$, and so on. For the (v, i) epoch, process v broadcasts the following information on a randomly chosen channel for each of the $\Theta(t^2 \log n)$ rounds: the message $m_{v,i}$, along with the *reconstruction hash* defined as $H_1(m_{v,i}, \ldots, m_{v,k})$. All other nodes simply choose a random channel on which to listen, and remember what they have received. By the end of each epoch (v, i), with high probability, every node has received v's transmission (as per a straightforward Chernoff bound and a union bound over the $n-1$ nodes). They may also have (potentially) received many spoofed messages.

Reconstruction Phase

The message gossip phase concludes with each node attempting to reconstruct M_v for each $v \in \Pi$. Having received $O(t^2 \log n)$ messages during each epoch for v, there may be exponentially many possible M_v vectors. The reconstruction hash allows us to reduces these to a polynomial number of possibilities. To perform the reconstruction for $v \in \Pi$, we arrange the received messages into *levels* $1, \ldots, k$. In each level i we place all messages received during v's i^{th} epoch. We will attempt to *decorate* these levels with directed edges. Each directed edge will connect messages at some level i to messages at level $i+1$.

The decoration proceeds backwards as follows, beginning with level k, the largest level: For each message m_1 at level $k-1$, and for each message m_2 at level k, calculate the hash of each sequence consisting of message m_1, followed by message m_2. Draw an edge from message m_1 to message m_2 if and only if the reconstruction hash tagged on message m_1 equals $H_1(m_1, m_2)$.

We repeat this procedure as we move down the levels. For each level i message m_1, consider the hash produced by each of the sequences consisting of a level $i+1$ message, followed by a chain of messages reached by following outgoing edges up to level k; create an edge between m_1 and the level $i+1$ message when the calculated hash matches the reconstruction hash attached to the level i message.

The procedure requires $O(t^4 \log^2 n)$ hashes for each of the $k-1$ levels. Assuming a collision-resilient hash function, each message will be decorated with a most one outgoing edge. (Recall, however, that local computation is relatively cheap compared to the cost of sending messages.) It follows that the resulting decorated levels will include at most $O(t^2 \log n)$ chains of messages, each described by a path from level 1 to level k. Among these, of course, will be one sequence that actually represents M_v.

The Vector Signature

In order to identify the correct sequence of messages and thus ensure authenticity, we use the f-AME protocol with the following modification: each message containing M_v is replaced with the constant-sized message $H_2(M_v)$, the *vector signature* for v. Whenever a node first receives the vector signature for v, it can compare it to the H_2 hashes of its $O(t^2 \log n)$ validly reconstructed M_v vectors from the reconstruction phase, and thus validate the single vector that is correct. Of course, once it knows the real M_v, it can extract any message from v required by f-AME. The authentication guarantee of f-AME ensures that the received vector signature is authentic.

6. A SHARED GROUP KEY

We now describe how to use f-AME to establish a secret key shared among all but at most t nodes that is unknown to the adversary. Group keys have a variety of uses from broadcast encryption to group management. In section Section 7, we describe one possible use in the setup of a long-lived, reliable, secure, and authenticated communication service.

Part 1: Initialize Shared Keys

We begin by choosing a set of $t+1$ *leader* nodes, and we establish a shared secret key between each leader node and each non-leader node. Let $\ell \subset \Pi$ contain $t+1$ *leader* nodes, and define the pair set $E_\ell = \{(v,w) \mid v \in \ell \vee w \in \ell\}$.

We then use f-AME to swap messages for a one-round key exchange protocol, e.g., Diffie-Hellman Key Exchange (DHKE) [12]. Specifically, execute f-AME where $E = E_\ell$ and each message $m_{v,w}$ is the appropriate DHKE message to be sent from v to w. Any non-disrupted process pair can use the DHKE messages to establish a shared key unknown to the adversary.[2] The total cost: $O(nt^3 \log n)$ rounds.

Part 2: Disseminate Leader Keys

Recall that f-AME guarantees *sender awareness*, and thus each leader knows how many shared secrets were successfully established. If a leader exchanges messages with at least $n-t$ nodes, it considers itself *complete*. Each complete leader chooses a *leader key* K_v.

We assign an *epoch* of length $\Theta(t \log n)$ for each pair (v,w) where $v \in \ell, w \in \Pi - \{v\}$. If v and w failed to establish a shared secret, they remain silent during their epoch. Otherwise, using their shared secret, they calculate a pseudo-random channel-hopping pattern that is unknown to the adversary. During each round of the epoch, if v is complete, it transmits the leader key K_v, encrypted with the key shared by v and w on the calculated pseudo-random channel. Otherwise, it sends the message "incomplete." Node w listens accordingly. The pair avoid interference in each round with probability at least $1/(t+1)$. Therefore, over $\Theta(t \log n)$ rounds, we conclude that w receives the leader key with high probability. The total cost: $\Theta(nt^2 \log n)$ rounds.

Part 3: Key Agreement

Finally, the nodes collectively choose one of the leader keys to be the shared group key. Let S be a collection of $2t+1$ non-leader nodes. We assign an epoch of length $\Theta(t^2 \log n)$ to each node $i \in S$. During this epoch, node i randomly chooses channels on which to broadcast the identity of the smallest leader j from which it received a key K_j during Part 2, and a hash of the key K_j. Other nodes receive on random channels. Because epochs lasts $\Theta(t^2 \log n)$ rounds, every node should hear from all nodes in S.

After the epochs conclude, we apply a simple agreement rule: If a node heard at least $t+1$ non-leader nodes report the same leader, and the node can verify each of these messages (i.e., it knows the leader key for this leader and can therefore confirm the leader key hash for these reports), then it adopts this leader key as the group key. Otherwise, it recognizes that it does not know the group key.

To see that this works, let j be the smallest completed leader. We know that at least $t+1$ nodes in S heard from j in Part 2, since j is complete, and thus report j in Part 3. (Recall that incomplete leaders do not broadcast in Part 2.) Therefore, the $n-t$ nodes that received j's leader key K_j confirm the reports and adopt K_j. (The other t nodes will correctly identify their lack of knowledge.) The total cost of this part: $\Theta(t^3 \log n)$ rounds. The total cost of the entire protocol is dominated by Part 1: $\Theta(nt^3 \log n)$ rounds. (With more channels, the cost can be reduced accordingly.)

7. LONG-LIVED COMMUNICATION

Once we have established shared secret keys, we can construct a long-lived communication protocol that simulates a secure channel. Thus nodes can execute the f-AME protocol once to bootstrap the secure channel emulation, and from then on out, rely on this more efficient and robust communication primitive. The long-lived communication protocol guarantees, with high probability: (1) **t-Reliability:** all but t nodes can communicate on the emulated channel; (2) **Secrecy:** the adversary cannot eavesdrop on the channel; and (3) **Authentication:** a node w receives a message m from v only if v previously sent message m.

Such a primitive is easy to implement using a shared secret group key K, as established in Section 6. The nodes generate a channel-hopping pattern by using K as the seed to a pseudo-random generator. The nodes then participate in each round on the specified channel.

To broadcast in an emulated round, a node repeats its message, encrypted using key K, for $\Theta(t \log n)$ rounds; other nodes simply listen. Since the channel-hopping appears random to the adversary, it is easy to see that if only one node broadcasts, its message will be received, without disruption, with high probability. (As before, for $C \geq 2t$, the number of required real rounds would fall to $O(\log n)$.) If two or more nodes broadcast concurrently, then, as on a real broadcast channel, one or more messages might be lost. The channel emulation requires $\Theta(nt^3 \log n)$ rounds to setup the group key, and each emulated round requires $\Theta(t \log n)$ rounds.

[2]The specific security assumption for DHKE holds that calculating the shared key can be reduced to the Computational Diffie-Hellman assumption; that is, calculating discrete log in a large finite field.

8. OPEN QUESTIONS

By focusing on multi-channel wireless communication, this paper introduces a new environment for the study of secure communication. Thus, there are a variety of open questions to consider. (1) Byzantine corruption: can we tolerate corruption faults, where the set of honest, participating devices is unknown? A simple modification allows us to achieve $2t$-disruptability in this case: surrogates are eliminated, and every rumor is received directly from its source; $t+1$ nodes are used to report on every non-disrupted channel during communication-feedback. However, it remains open whether t-disruptability can be achieved. (2) Information-theoretic security: the secrecy of the shared keys here depends on cryptographic assumptions, such as (computational Diffie-Hellman). We can restrict the adversary to listening on only t channels, so at least one channel's communication is unknown. In this case, is it possible to achieve some shared secrets that are secure in an information-theoretic sense? We conjecture that any such algorithm in inherently exponential in time complexity. (3) Beyond t-disruptability: is it possible to make some progress with the disrupted nodes, even if it is at the cost of weakening, for them, some of the AME guarantees? (4) Point-to-point communication and beyond: do there exist more efficient point-to-point primitives? What other communication and coordination activities are practical? (5) Multi-hop networks: all of these questions can be studied in the context of multi-hop networks.

9. REFERENCES

[1] N. Alon, A. Bar-Noy, N. Linial, and D. Peleg. A lower bound for radio broadcast. *Journal of Computer and System Sciences*, 43(2):290–298, October 1992.

[2] R. Bar-Yehuda, O. Goldreich, and A. Itai. On the time-complexity of broadcast in multi-hop radio networks: An exponential gap between determinism and randomization. *Journal of Computer and System Sciences*, 45(1):104–126, 1992.

[3] R. Bar-Yehuda, A. Israeli, and A. Itai. Multiple communication in multi-hop radio networks. *SIAM Journal on Computing*, 22(4):875–887, 1993.

[4] V. Bhandari and N. H. Vaidya. On reliable broadcast in a radio network. In the *Proceedings of the International Symposium on Principles of Distributed Computing*, pages 138–147, July 2005.

[5] Bluetooth Consortium. *Bluetooth Specification Version 2.1*, July 2007.

[6] M. Cagalj, S. Capkun, and J-P. Hubaux. Key agreement in peer-to-peer wireless networks. *Proceedings of the IEEE* (Special Issue on Cryptography and Security), 94(2), February 2006.

[7] B. S. Chlebus, L. Gasieniec, A. Lingas, and A. T. Pagourtzis. Oblivious gossiping in ad-hoc radio networks. In the *Proceedings of the 5th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 44–51, 2001.

[8] B.S. Chlebus and D.R. Kowalski. Robust gossiping with an application to consensus. *Journal of Computer and System Sciences*, 72(8):1262–1281, December 2006.

[9] A. Clementi, A. Monti, and R. Silvestri. Optimal f-reliable protocols for the do-all problem on single-hop wireless networks. *Algorithms and Computation*, pages 320–331, 2002.

[10] A. Clementi, A. Monti, and R. Silvestri. Round robin is optimal for fault-tolerant broadcasting on wireless networks. *Journal of Parallel and Distributed Computing*, 64(1):89–96, 2004.

[11] Artur Czumaj and Wojciech Rytter. Broadcasting algorithms in radio networks with unknown topology. In the *Proceedings of the Symposium on Foundations of Computer Science*, October 2003.

[12] W. Diffe and M. Hellman. New directions in cyptography. *Transactions on Information Theory*, November 1976.

[13] S. Dolev, S. Gilbert, R. Guerraoui, and C. Newport. Gossiping in a multi-channel radio network: An oblivious approach to coping with malicious interference. In the *Proceedings of the International Symposium on Distributed Computing*, September 2007.

[14] S. Gilbert, R. Guerraoui, and C. Newport. Of malicious motes and suspicious sensors: On the efficiency of malicious interference in wireless networks. In the *Proceedings of the International Conference on Principles of Distributed Systems*, December 2006.

[15] J. Komlos and A.G. Greenberg. An asymptotically fast non-adaptive algorithm for conflict resolution in multiple access channels. *IEEE Transactions of Information Theory*, 31(2,) March 1985.

[16] C-Y. Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. In the *Proceedings of the International Symposium on Principles of Distributed Computing*, pages 275–282, July 2004.

[17] C-Y. Koo, V. Bhandari, J. Katz, and N. H. Vaidya. Reliable broadcast in radio networks: The bounded collision case. In the *Proceedings of the International Symposium on Principles of Distributed Computing*, July 2006.

[18] D. Kowalski and A. Pelc. Time of deterministic broadcasting in radio networks with local knowledge. *SIAM Journal on Computing*, 33(4):870–891, 2004.

[19] E. Kranakis, D. Krizanc, and A. Pelc. Fault-tolerant broadcasting in radio networks. *Journal of Algorithms*, 39(1):47–67, April 2001.

[20] E. Kushelevitz and Y. Mansour. An $\Omega(d \log(n/d))$ lower bound for broadcast in radio networks. In the *Proceedings of the International Symposium on Principles of Distributed Computing*, August 1993.

[21] A. Pelc and D. Peleg. Feasibility and complexity of broadcasting with random transmission failures. In the *Proceedings of the International Symposium on Principles of Distributed Computing*, pages 334–341, July 2005.

[22] F. Stajano and R. Anderson. The resurrecting duckling: security issues for ad hoc wireless networks. In the *Proceedings of the International Workshop on Security Protocols*, April 1999.

[23] D. E. Willard. Log-logarithmic selection resolution protocols in a multiple access channel. *SIAM Journal of Computing*, 15(2):468–477, 1986.