A System-Theoretic Clean Slate Approach to Provably Secure Ad Hoc Wireless Networking

Jonathan Ponniah*, *Member, IEEE,* Yih-Chun Hu*, *Member, IEEE,* and P. R. Kumar**, *Fellow, IEEE*

***Abstract*—Traditionally, wireless network protocols have been designed for performance. Subsequently, as attacks have been identified, patches have been developed. This has resulted in an "arms race" development process of discovering vulnerabilities and then patching them. The fundamental difficulty with this approach is that other vulnerabilities may still exist. No provable security or performance guarantees can ever be provided.**

We develop a system-theoretic approach to security that provides a complete protocol suite with provable guarantees, as well as proof of min-max optimality with respect to any given utility function of source-destination rates. Our approach is based on a model capturing the essential features of an ad-hoc wireless network that has been infiltrated with hostile nodes. We consider any collection of nodes, some good and some bad, possessing specified capabilities vis-a-vis cryptography, wireless communication and clocks. The good nodes do not know the bad nodes. The bad nodes can collaborate perfectly, and are capable of any disruptive acts ranging from simply jamming to non-cooperation with the protocols in any manner they please.

The protocol suite caters to the complete life-cycle, all the way from birth of nodes, through all phases of ad hoc network formation, leading to an optimized network carrying data reliably. It provably achieves the min-max of the utility function, where the max is over all protocol suites published and followed by the good nodes, while the min is over all Byzantine behaviors of the bad nodes. Under the protocol suite, the bad nodes do not benefit from any actions other than jamming or cooperating.

This approach supersedes much previous work that deals with several types of attacks including wormhole, rushing, partial deafness, routing loops, routing black holes, routing gray holes, and network partition attacks.

***Index Terms*—Ad hoc wireless networks, security.**

I. Introduction

OUR focus is on the problem of security of ad-hoc, multi-hop, wireless networks. The wireless nodes in these types of networks need to determine when to transmit packets and at what power levels, discover routes from sources to destinations, and ensure overall end-to-end reliability, all without any centralized controller guiding the process. This requires a suite consisting of multiple protocols.

Several candidates have been proposed. Medium access control protocols include IEEE 802.11 [11] and MACAW [2], power control protocols include COMPOW [13] and PCMA [16], routing protocols include DSDV [19], AODV [18], DSR [12], and OLSR [5], and transport protocols include TCP [22] and variations for ad hoc networks [14], [7], [3], [23].

*CSL & ECE, Univ. of Illinois, 1308 West Main St., Urbana, IL 61801. Email: {ponniah1,yihchun}@illinois.edu. Tel. 217-333-4220.

**Corresponding author: ECE, Texas A&M University, 3259 TAMU, College Station, TX 77843-3259. Email: prk@tamu.edu. Tel: 979-862-3376.

This paper is partially based on work supported by NSF under Contract Nos. CNS-1302182, CCF-0939370 and CNS-1232602, AFOSR under Contract No. FA-9550-13-1-0008, and USARO under Contract No. W911NF-08-1-0238.

All the above protocols are designed on the assumption that all nodes are "good," and will conform to the protocol. Some nodes can however be malicious, deliberately intent on disrupting the network, a vulnerability especially acute since the very purpose of ad hoc networks is to allow any node to join a network. For wireless networks used in safety-critical applications, e.g., vehicular networks, vulnerabilities can be dangerous. Moreover, many wireless networking protocols have been based on wireline protocols, with possible susceptibilities to novel over the air attacks.

The assumption of benignness, implicit or explicit, has been the traditional starting point of protocol development. Systems have been first designed to provide high performance. Subsequently, as vulnerabilities have been discovered, they have been patched on a case by case basis. For example, the "wormhole" attack was discovered in [8], where an attacker sets up a false link between two nodes. It is countered by a fix using temporal and geographical packet leashes [8], [21]. The "rushing" attack against DSR was discovered in [9], in which attackers manipulate the network topology. This is countered by a fix using network discovery chains. The "partial deafness" attack against 802.11 was discovered in [4], in which an attacker artificially reduces its link quality to draw more network resources. It is countered by a fix using queue regulation at the access point. Other attacks against DSR are the routing loop attack in which an attacker generates forged routing packets causing data packets to cycle endlessly; the routing black hole attack in which an attacker simply drops all packets it receives; and the network partition attack in which an attacker injects forged routing packets to prevent one set of nodes from reaching another. These attacks are all countered in the Ariadne protocol [10] by the joint use of routing chains, encryption, and packet leashes. Some protocols such as Watchdog and Pathrater [15] try to pre-empt attacks by maintaining a blacklist that tracks malicious behavior, but this backfires if an attacker maligns a good node, causing other good nodes to add that node to their blacklists. These attacks are not targeted at violating privacy of communications between nodes, which can be avoided simply by encryption. Rather, they are generally Denial of Service attacks (DoS), which usually take advantage of algorithms that assume the participating users are good or cooperative.

The basic problem with this arms race approach of hardening algorithms initially designed for good performance is that one never knows what other vulnerabilities or attacks exist. Thus no guarantees can be provided about the security of the protocols at any stage of the arms race process.

Our goal is to propose an alternate clean slate system-theoretic approach to security that provides provable performance guarantees. We pursue a model-based approach, comprising a physical model of node capabilities, clocks, cryptography, and wireless communication. It is an initial attempt to holistically model the entire dynamics of an ad-hoc wireless network that has been infiltrated with hostile nodes.

Our goal is to design a protocol suite for the complete life-cycle of the wireless system, all the way from the very birth of the nodes, and continuing through all phases of the network formation process, to a long-term operation where the network is carrying data reliably from sources to their destinations. The good nodes don't know who the bad nodes are, and are required to follow the published protocol suite. Throughout all phases, the bad nodes can perfectly collaborate and incessantly indulge in any disruptive behavior to make the network formation and operation dysfunctional. They could just "jam," or engage in more intricate behavior such as not relay a packet, advertise a wrong hop count, advertise a wrong logical topology, cause packet collisions, disrupt attempts at cooperative scheduling, drop an ACK, refuse to acknowledge a neighbor's handshake, or behave inconsistently.

We design a protocol suite that is provably secure against all such attacks by the malicious nodes. Not only that, it guarantees min-max optimal performance, described by a given utility function. The good nodes maximize it by publishing a complete protocol suite and conforming to it, while the bad nodes minimize it by indulging in all manner of "Byzantine" behavior described above not conforming to the protocol.

This leads to a zero-sum game. Since the good nodes first announce the protocol, the best value of the utility function that the good nodes can hope to attain is its max-min, where the maximization is over all protocol suites, and the minimization is over all Byzantine behaviors of the bad nodes. We will prove that the protocol suite designed attains this max-min to within any $\epsilon > 0$. Moreover, we establish three even stronger results.

First, this game actually has a saddle point, i.e., the protocol suite attains the min-max (to within any $\epsilon > 0$). (Generally, min-max results in a higher utility than max-min, since the bad nodes have to first disclose their tactics).

Second, the bad nodes can do no better than just jamming or conforming to the published protocol suite on each "concurrent transmission vector," a generalization of the notion of an "independent set" of nodes that can simultaneously transmit. They do not benefit from more elaborate Byzantine antics.

Third, If the bad nodes behave suboptimally, i.e., are not totally hostile, the protocol optimally exploits it. This is a desirable feature since one would want them to exploit any benignness in the environment.

Some important qualifications need to be noted. First, the results are valid only for the postulated model of the network. Future research may identify technological capabilities outside the model that can attack the protocol suite. Such discoveries will, one hopes, lead to the development of more general models and procotols provably secure in them. The research enterprise will thereby be elevated to a higher level; instead of reacting to each proposed protocol one reacts to each proposed model, with provable guarantees provided at each step. Section VII provides some such directions for model generalization.

Second, the optimality is over a large time period, and the overhead of transient phases of the protocol may be high. However, there is much scope for optimizing protocol overhead while preserving security.

Third, how should one view the proposed protocol suite? The answer is layered. At a minimum, it can be regarded as a constructive existence proof that one can indeed provide optimal performance while guaranteeing security, with the identified model class only serving as an exemplar of conditions under which this can be done. To a more receptive reader, the designed protocol suite is suggestive of how one can do so. The architectural decomposition into several phases could perhaps be kept in mind by future protocol designers.

At any rate, we hope that this approach will trigger several critical reactions among a skeptical readership, and lead to follow up work that designs better protocols with guaranteed security and performance for better models.

II. The Model

The model of an ad-hoc wireless network infiltrated by hostile nodes can be organized into four categories: the nodal model **(N)**, communication model **(CO)**, clock behavior **(CL)**, and cryptographic capabilities **(CR)**.

Nodal model: (N1) There are n nodes, some good and some bad. Let G denote the set of good nodes, and its complement B the set of bad nodes. **(N2)** The good nodes do not know who the bad nodes are a priori. **(N3)** The bad nodes are able to fully coordinate their actions, and are fully aware of their collective states (equivalent to unlimited bandwidth with zero delay between them). **(N4)** The good nodes are all initially powered off, and they all turn on within U_0 time units of the first good node that turns on.

Communication model: We consider an abstraction of a communication model that is fully deterministic. **(CO1)** Each node i can choose from among a finite set of transmission/reception modes $\mathcal{M}_i$ at each time. Each mode corresponds, if transmitting, to a joint choice of power level, modulation scheme and encoding scheme for each other intended receiver node, or to just listening and not transmitting, or even to a distinguished mode called "jamming" (which models using its power output to emit noise). **(CO2)** The good nodes are half-duplex, i.e., cannot transmit and receive simultaneously. **(CO3)** We call $c = (c_1, c_2, \ldots, c_n)$ denoting the mode choices of all the nodes made at a certain time, as a "concurrent transmission vector" (CTV). (It is more general than an independent set that is sometimes used to model wireless networks). We will denote by $c_G = (c_i : i \in G)$ and $c_B = (c_i : i \in B)$ the vectors of choices of modes made by the good and bad nodes respectively, with each $c_i \in \mathcal{M}_i$, and let $\mathcal{C}_G$ and $\mathcal{C}_B$ denote the sets of all such choices. We will denote by $\mathcal{C} := \mathcal{C}_G \times \mathcal{C}_B$, the set of all CTVs. **(CO4)** Each c results in a "link-rate vector" $r(c)$ of dimension $n(n-1)$. Its ij-th component, $r_{ij}(c)$, is the data rate at which bits sent by node i can be decoded at node j at that time. Due to the shared nature of the wireless medium, the rate depends on the transmission mode choices made by all the other nodes, as well as the geographic locations of the nodes, the propagation path loss, the ambient noise, and all other physical characteristics affecting data rate. A component $r_{ij}(c)$ may be zero, for example if the SINR at j is below a threshold value for decoding, or if node i is not transmitting to node j. If a single node changes its mode to "jam," the resulting rate vector is coordinate wise no larger than what it is when that node chooses any other mode. **(CO5)** If a certain rate vector is achievable then lower rates are also achievable. To state this, define $\mathcal{A}_{ij} := \{r_{ij}(c) : c \in \mathcal{C}\}$, and let $\mathcal{A} := \cup_{i \neq j} A_{ij}$denote the finite set of all possible nonnegative rates that can be achieved, with $0 \in \mathcal{A}$.[1] We suppose that for every c, and $r' \leq r(c)$ (understood component wise) with all elements in $\mathcal{A}$, there is a choice $c' \in \mathcal{C}$ such that $r(c') = r'$. **(CO6)** In the case of a bad node j, the rate $r_{ij}(c)$ may be the result of some other bad node being able to decode the packet from i at that rate, and then passing on that packet to j, since bad nodes can collaborate perfectly. In the case of a bad node i, the rate $r_{ij}(c)$ may be the result of some other bad node being able to transmit the packet successfully to j at that rate, pretending to be i. Meanwhile, in either case, the bad node may be jamming. Thus a bad node can both jam and appear to be cooperating, whether transmitting or receiving, at the same time. **(CO7)** The bad nodes can claim to have received transmissions from each other at any of the rates in the finite set $\mathcal{A}$, as they please. To state this, for $c = (c_G, c_B)$, we will partition the resulting link-rate vector as $r(c) = (r_{GG}(c), r_{GB}(c), r_{BG}(c), r_{BB}(c))$, where r_{BG} denotes the link-rates from the bad nodes to the good nodes, etc. We suppose that for every $c = (c_G, c_B)$ and every r' with all elements in $\mathcal{A}$, there is a $c'_B \in \mathcal{C}_B$ such that $r(c_G, c'_B) = (r_{GG}(c), r_{GB}(c), r_{BG}(c), r')$. with all elements in $\mathcal{A}$, **(CO8)** The good nodes know $\mathcal{A}$, and an upper bound on the cardinalities of the $\mathcal{M}_i$'s, but do not know the values of the vectors $r(c)$ for any $c \in \mathcal{C}$, and an upper bound on the cardinalities of the $\mathcal{M}_i$'s, but do not know the values of the vectors $r(c)$ for any $c \in \mathcal{C}$. **(CO9)** The assumption that the link-rate vector $r(c)$ does not change with time implicitly assumes that nodes are not mobile to any significant extent.

Clock model: (CL1) Each good node i has a local continuous-time clock that it initializes to zero when it turns on. Its time $\tau^i(t)$ is affine with respect to some reference time $t \geq 0$, i.e., $\tau^i(t) = a_i t + b_i$ where a_i and b_i are called the skew and offset respectively. Wlog, the time t above and in (N4) is taken equal to the clock time of the first good node to turn on. **(CL2)** Denoting the relative skew and offset between nodes i and j by $a_{ij} := \frac{a_i}{a_j}$ and $b_{ij} := b_i - a_{ij} b_j$, node i's time with respect to node j's time s is $\tau^i_j(s) = a_{ij} s + b_{ij}$. We assume $0 < a_{ij} \leq a_{max}$. As a corollary of (N4,CL1,CL2), $|b_{ij}| \leq a_{max} U_0$, since $\tau^i(U_0) \geq 0$. **(CL3)** The good nodes do not know their skew or offset a priori. **(CL4)** Finally, due to its digital processor, a good node i can only observe a quantized version of its continuous-time local clock $\tau^i(t)$.

Cryptographic capabilities: (CR1) Each node is assigned a public key and a private key; information encrypted by a public key can only be decrypted with the corresponding private key. A message signed by a private key can be verified by any node with the public key. The private key is never revealed by a good node to any other node. Possession of a public key does not enable an attacker to forge, alter, or tamper with an encrypted packet generated with the corresponding private key. The good nodes encrypt all their transmissions. **(CR2)** Each node possesses the public key of a central authority. **(CR3)** Each node possesses an identity certificate, signed by the central authority, containing node i's public key and ID number. The certificate binds node i's public key to its identity. **(CR4)** Each node possesses a list of all the other n node IDs.

Now we describe a key "connectedness assumption" that makes cooperation among the good nodes feasible even in the face of hostile Byzantine attacks by the bad nodes. Consider a mode $c = (c_G, c_B)$. It attains a rate $r(c_G, c_B)$. Now suppose that the protocol at some point requires the implementation of (c_G, c_B). Towards this end, the good nodes would faithfully choose their portion c_G. Now suppose that there exists a choice c'_B such that for some link (i,j), $r_{ij}(c_G, c'_B) < r_{ij}(c_G, c_B)$. That is, by their actions, the bad nodes can prevent the intended rate from being realized, at least for one link. Then we say that the CTV $c = (c_G, c_B)$ *can be disabled* by the bad nodes.[2] Let $\mathcal{D}^*$ denote the set of all such CTVs that can be disabled. Let $\mathcal{G}^*$ be defined as the directed graph over the nodes, where there is an edge ij if and only if $r_{ij}(c) > 0$ for some $c \notin \mathcal{D}^*$. This is the graph over which communication cannot be disrupted by the bad nodes.

Connectedness Assumption (C): We will assume that the set consisting of all good nodes is connected in the subgraph of $\mathcal{G}^*$ that consists only of edges ij for which both ij as well as ji are edges in $\mathcal{G}^*$.

We now consider a utility function of the throughputs of any subset S of source-destination pairs. We define it for all subsets since we will consider the utility of those nodes that are perceived as conforming to the protocol.

Utility function assumption (U): For any subset $S \subseteq \{1, 2, \ldots, n\}$ and any $n(n-1)$-dimensional end-to-end throughput vector x, let $U(x, S)$ depend only on x_{ij} for $i, j \in S$. For every S, $U(x, S)$ is continuous and monotone increasing in the components of x.

In order to quantify how good a performance is achievable, we need to specify what end-to-end rates are actually feasible through multi-hop. Suppose that $\mathcal{D} \subset \mathcal{D}^*$ denotes the particular set of CTVs that the bad nodes has chosen to disable. Let us call the complement $\mathcal{E} := \mathcal{C} \setminus \mathcal{D}$ as the "allowed set." Let $R(\mathcal{E}) := \text{ConvexHull}(\{r(c) : c \in \mathcal{E}\})$ *be the set of link rate-vectors supported by* $\mathcal{E}$ *by time-sharing* over the CTVs in $\mathcal{E}$. Denote by P_{ij} the set of all multi-hop paths from i to j. We can now define the *multi-hop capacity region* of $n(n-1)$-dimensional end-to-end source-destination throughput vectors in the standard way as $C(\mathcal{E}) := \{x :$ For some vector $y \geq 0$ with $0 \leq \sum_{p:\ell \in p} y_p \leq r_\ell$ for some

[1]For example, $\mathcal{A} = \{0, 6, 9, 12, 18, 24, 36, 48, 54\}$ (Mbps) in IEEE 802.11a, with 6Mbps achieved by BPSK at code rate 1/2, 9Mbps achieved by BPSK at code rate 3/4, 12 achieved by QPSK at code rate 1/2, 18 achieved by BPSK at code rate 3/4, 24 achieved by 16-QAM at code rate 1/2, 36 achieved by BPSK at code rate 3/4, 48 achieved by 64-QAM at code rate 1/2, 54 achieved by BPSK at code rate 1/2.

[2]As an example, if a bad node jams, it can prevent a rate vector from being realized. Thus it "disables" the particular CTV. As another example, if a bad node reports that it did not receive a packet sent at a certain target rate. Then again it effectively disables that particular CTV.

$r \in R(\mathcal{E})$, $x_{ij} = \sum_{p \in P_{ij}} y_p$ for all $1 \leq i,j \leq n, j \neq i\}$. We also define $\mathcal{G}(\mathcal{E})$ as the directed graph over the nodes, where there is an edge ij if and only if $r_{ij}(c) > 0$ for some $c \in \mathcal{E}$.

We now consider the game where the good nodes wish to maximize the utility for the nodes perceived to be good by them, while the bad nodes wish to minimize it over all their Byzantine behaviors. To obtain an upper bound on utility, suppose that the bad nodes disable only the CTVs in $\mathcal{D}$ and reveal this choice to the good nodes. If $\mathcal{G}(\mathcal{E})$ has several strongly connected components, then, by the connectedness assumption (C), the good nodes are all in the same component, denoted by $F(\mathcal{E})$, and thus know that the nodes outside $F(\mathcal{E})$ are bad. They will therefore only consider the utility accrued as $U(x, F(\mathcal{E}))$, and maximize it over all $x \in C(\mathcal{E})$. An upper bound on this achievable utility is therefore $\min\limits_{\mathcal{D} \subset \mathcal{D}^*} \max\limits_{x \in C(\mathcal{C} \setminus \mathcal{D})} U(x, F(\mathcal{C} \setminus \mathcal{D}))$.

III. The Outline of the Approach

We will show in this paper that, in spite of perfect collusion and collaboration between all the bad nodes in engaging in all manner of Byzantine behaviors, such as, but not limited to, not relaying a packet, advertising a wrong hop count, advertising a wrong logical topology, behaving uncooperatively vis-a-vis medium access, disrupting attempts at cooperative scheduling, dropping ACKs, refusing to acknowledge a neighbor's handshake, behaving inconsistently, and given that the good nodes have to build a complete network from scratch right from birth without knowing who their neighbors are, and without the good nodes knowing the identity of the bad nodes, the good nodes can achieve the above optimized utility arbitrarily closely. Moreover, the fact that it is the min-max and not just the max-min that is achievable, has several implications. Our proof is constructive, providing a complete protocol suite from birth to network formation and operation of a utility optimizing network.

Our scheme above relies heavily on the Byzantine agreement algorithm. This allows the nodes to agree on a common view, even in the presence of malicious nodes. Additionally, we go beyond mere Byzantine agreement and ensure that the common view is actually internally consistent.[3] Subsequently, after agreeing on a common view, which includes clocks, each node can act in a distributed fashion, with no need to exchange control messages, This enables our protocol to withstand any malicious attacks on control packets. Since the malicious nodes cannot attack control messages, they are left with only attacking data messages. That is, they can only jam if they want to. This explains why we are able to obtain the min-max optimality result.

Our agreement on a common view includes an extension of what one may call "logical topology," as well as clock information. The extension of logical topology is necessary because a wireless network cannot simply be described as a graph with rates on links, as wireline networks can. In a wireless network, whether a pair of nodes can communicate depends on what other nodes are concurrently transmitting and at what power, as well as whether the recipient node is itself "listening," since we allow the good modes to be only half-duplex. Thus what happens in a wireless network depends on the nature of all the concurrent transmissions. So our Byzantine agreement is about the rate vectors achieved by various possibilities for concurrent transmissions, i.e., what we have called "CTVs" above.

Our Byzantine agreement is also about the clocks at different nodes. Once that is done nodes can cooperate simply by the times of their own clocks.[4] Hence they will not need further control messages to cooperate.

The heart of the approach after agreeing on a common view is to investigate different CTVs, exploiting the fact that the operation of the network consists of invoking which particular CTV to use at any given instant. If a good node fails to receive a scheduled packet, then that good node alerts the rest of the network during a verification phase, and the offending CTV set is never used again. After each such pruning the network then re-optimizes its utility over the remaining CTVs. [5] The decreasing sequence of residual remaining sets of CTVs necessarily converges to an operational collection of CTVs, over which the utility is optimized by time sharing. Since the set of disabled CTVs is determinable by the network, as we show, it is the same as if it were revealed to the good nodes a priori, which allows achievement of min-max. It also shows that more complex Byzantine behaviors than jamming or cooperating are not any more effective for the bad nodes.

There are however several problems that lie along the way to realizing this scheme. How does one determine the network topology and other parameters, while under attack from bad nodes? During this period even control packets can be attacked. We present a complete protocol suite that proceeds through several phases to achieve this end result. After their birth, the nodes need to first discover who their neighbors are. This requires a two-way handshake, which presents one problem already. Two good nodes that are neighbors can successfully send packets to each other if there are no primary (half-duplex) or secondary (collision) conflicts. To achieve this we employ an Orthogonal MAC Code [20].[6] Next, the two nodes need to update their clock parameters. After this, the nodes propagate their neighborhood information so that everyone learns about the network topology. This also poses some challenges when there are intermediary bad nodes. This is addressed by a version of the Byzantine General's algorithm

[3] For example, we ensure that the product of the clock skews along, say, one path node 1 to node 2 to node 3 is the same as the product of the clock skews along, say, another path node 1 to node 4 to node 2.

[4] This statement is more subtle than it sounds since it turns out that it is impossible for nodes to ever determine each other's clocks [6] whenever there is asymmetric delay. However they can successfully time a packet transmission to arrive at the intended node at a certain time according to that node's cock, which is all that the protocol needs.

[5] The verification phase, as indeed all else, uses Byzantine agreement algorithm. Similarly, agreement is reached via the Byzantine agreement algorithm after each pruning of what CTVs appear to be working. Thus the Byzantine agreement algorithm is invoked multiple times at each pruning of the set of CTVs to recursively generate common views.

[6] This is a code that specifies windows for each node according to their own clocks when they should transmit and to whom, and when should just listen. These times are so chosen that within a bounded time every pair of nodes can exchange packets, even prior to clock synchronization.

of [1]. This paper proved both agreement and validity for graphs in which the number of faulty nodes f is no more than a third of the total, that is, $n > 3f + 1$, and the connectivity of the graph G satisfies $conn(G) > 2f$. However, it is also well established that agreement and validity hold in complete graphs for $n > f$ if the network employs perfect authentication [17]. This result can also be extended to graphs such as ours, in which the legitimate nodes are connected. Next, even though all the good nodes converge to a common network view, that view may be internally inconsistent, especially with respect to clocks. To resolve this we employ a certain consistency check algorithm. Next, the nodes proceed to determine an optimal schedule for time sharing over the set of CTVs that have performed consistently from the very beginning, and execute it. However, a bad node that has cooperated hitherto may not cooperate at this point. Hence the results of this operational phase need to be verified, the dysfunctional CTV pruned, the schedule re-optimized, and the procedure iterated.

The reader may wonder: Why do we even need a notion of "time"? First, without it, we cannot even speak of throughput or thus of utility. Second, we use local clocks to schedule transmissions and coordinate activity (as is quite common, e.g., time-outs in MAC and transport protocols). On the other hand, dependence on distributed synchronized clocks for coordinated activity opens yet another avenue for bad nodes to sabotage the protocol – interfering with the clock synchronization algorithm. Therefore, topics like scheduling, clock synchronization, utility maximization, are deeply interwoven. Therefore one needs a holistic approach that addresses all these issues at every stage of the operating lifetime, and further guarantees min-max optimality.

One obvious drawback of our approach is the complexity of Byzantine agreement. Concerning this, there are several points to be kept in mind. First, technically speaking, we amortize this overhead by running a very long data transmission and thus utility accruing phase. That is why we are able to prove ϵ-min-max optimality. Nevertheless the transients are unrealistically long if one implements our algorithm exactly the way specified. However, one can optimize on this, and we hope that follow up papers will do so, for example by postponing verification until all data is exchanged, since it may well be the case that there are no malicious nodes. Thus one may be able to design lower complexity algorithms where complex algorithms are only invoked if necessary, but otherwise nodes proceed optimistically. All of the above is in line with our dictum that one can optimize algorithms under ultimate security guarantees. Finally, it must be kept in mind that this paper is only intended to be an initial foray to show that provable security is achieved under appropriate models and how to do so, and also that we are hoping that others will propose alternative models that are both more realistic as well as more tractable. Some important further desirable directions for extension are outlined in Section VII.

This is the raison d'être for this paper.

IV. The Phases of the Protocol Suite

The protocol suite we show to be optimal consists of six phases: Neighbor Discovery, Network Discovery, Consistency Check, Scheduling, Data Transfer, and Verification.

We first note the necessity for a key ingredient, guaranteed medium access control. Even two good nodes that are neighbors as in assumption (C) are only guaranteed to be able to successfully send packets to each other provided one is transmitting, the other is listening (since good nodes are half-duplex), and the remaining good nodes are all silent. The Orthogonal MAC Code (OMC) of [20] ensures the simultaneity of all these events, even though the clocks of different nodes have different skews and offsets. For each pair of nodes i, j, it defines certain zero-one valued functions of local time at each node, such that if i transmits a packet of duration W to j at that time, then the packet is successfully received, and the delay involved in waiting for such an eventuality is never more than a certain $T_{MAC}(W)$.

A. The Neighbor Discovery Phase

In this phase, each node i will determine the identity and relative clock parameters of nodes in its neighborhood $\mathcal{N}_i$, and include this data in a mutually authenticated link certificate.

In the first two steps, each node i attempts a handshake with a neighbor node j by broadcasting a probe packet PRB_{ij} and waiting for an acknowledgement ACK_{ji}. The probe packet contains an identity certificate signed by a central authority. Given $\mathcal{N}_i := \{1, \ldots, n\} \backslash i$, an initial candidate for the set of bidirectional neighbors of i (as in (C)), to indicate that node i transmits PRB_{ij} to each node $j \in \mathcal{N}_i$ via the OMC, and receives PRB_{jj} from each node $j \in \mathcal{N}_i$, we use TxRxMAC($PRB_{i \to \mathcal{N}_i}$,$PRB_{\mathcal{N}_i \to i}$). If a probe packet is not received from some node j, then j is pruned from $\mathcal{N}_i$.

Next, node i transmits an acknowledgment ACK_{ij} to node j containing a signed confirmation of the received probe packet PRB_j. Node i also listens for an acknowledgment ACK_{ji} from node j. Node i further removes from $\mathcal{N}_i$ any nodes that failed to return acknowledgements.

Then node i transmits to each node $j \in \mathcal{N}_i$ a pair of timing packets $TIM_{i,j}^{(1)}$ and $TIM_{i,j}^{(2)}$ that contain the send-times $s_{ij}^{(1)}$ and $s_{ij}^{(2)}$ respectively as recorded by its local clock $\tau^i(t)$. Node i also receives a corresponding pair of timing packets $TIM_{j,i}^{(1)}$ and $TIM_{j,i}^{(2)}$ from node j, and records the corresponding receive-times $r_{ji}^{(1)}$ and $r_{ji}^{(2)}$ respectively, as measured by the local clock $\tau^i(t)$. Any node that fails to deliver timing packets to node i is further removed from $\mathcal{N}_i$. The timing packets are used to estimate the relative skew a_{ji} by $\hat{a}_{ji} := \frac{r_{ji}^{(2)} - r_{ji}^{(1)}}{s_{ji}^{(2)} - s_{ji}^{(1)}}$.

The relative skew is used at the end of the Network Discovery Phase, to estimate a reference clock with respect to the local continuous-time clock. In the last two steps, node i creates a link certificate $LNK_{ij}^{(1)}$ containing the computed relative clock skew with respect to node j, and transmits this link to node j using the OMC. Node i also listens for a similar link certificate $LNK_{ji}^{(1)}$ from node j.

Finally, node i verifies and signs the received link certificate, and transmits the authenticated version $LNK_{ji}^{(2)}$ back to node j. Node i listens for a similar authenticated link certificate $LNK_{ij}^{(2)}$ from j. Any nodes that fail to return link certificates are removed from the set $\mathcal{N}_i$. This set now represents the nodes in the neighborhood of node i with whom node i has established mutually authenticated link certificates. The Neighbor Discovery Phase's pseudocode is shown in Algorithm 1.

One problem is that the algorithm must be completed in a partially coordinated manner even though the nodes are asynchronous; the completion of any stage in the Exponential Information Gathering (EIG) algorithm (see below) depends on the successful completion of the previous stages by all other good nodes. Consequently, we assign increasingly larger intervals $S_k := [t_k, t_{k+1}), k = 1, \dots 6$, to each successive protocol stage; see Section VI.

Algorithm 1 The Neighbor Discovery Phase

```
procedure NeighborDiscovery
  N_i := {1, ..., n} \ i
  while t ∈ S_1 do
    TxRxMAC(PRB_{i→N_i}, PRB_{N_i→i})
    Update(N_i)
  end while
  while t ∈ S_2 do
    TxRxMAC(ACK_{i→N_i}, ACK_{N_i→i})
  end while
  while t ∈ S_3 do
    TxRxMAC(TIM^(1)_{i→N_i}, TIM^(1)_{N_i→i})
    Update(N_i)
  end while
  while t ∈ S_4 do
    TxRxMAC(TIM^(2)_{i→N_i}, TIM^(2)_{N_i→i})
    Update(N_i)
  end while
  while t ∈ S_5 do
    TxRxMAC(LNK^(1)_{i→N_i}, LNK^(1)_{N_i→i})
    Update(N_i)
  end while
  while t ∈ S_6 do
    TxRxMAC(LNK^(2)_{i→N_i}, LNK^(2)_{N_i→i})
    Update(N_i)
  end while
end procedure
```

B. The Network Discovery Phase

The purpose of this Phase is to allow the good nodes to obtain a *common* view of the network topology and *consistent* estimates of all clock parameters. To accomplish this, the good nodes must disseminate their lists of neighbors to all nodes, so that all can decide on the same topology view. However good nodes do not know a priori which nodes are bad, and so bad nodes can selectively drop lists or introduce false lists to prevent consensus. We resolve this by using a version of the Byzantine General's algorithm of [1], requiring an EIG tree data structure. Let T_i denote node i's EIG tree, which by construction has depth n. The root of T_i is labelled with node i's neighborhood, i.e., the nodes in $\mathcal{N}_i$ and the corresponding collection of link certificates. First node i transmits to every node $j \in \mathcal{N}_j$ in its neighborhood, the list of nodes in $\mathcal{N}_i$ and corresponding link certificates, while receiving similar lists from each node in $\mathcal{N}_j$. Node i updates its EIG tree with the newly received lists from its neighbors, by assigning each received list to a unique child vertex of the root of T_i. Node i then transmits the set of level 1 vertices of T_i to every node in its neighborhood, receiving a set of level 1 vertcies from each neighbor in turn. The EIG tree T_i is updated again. This process continues through all n levels of the EIG tree.

The notation $T_i^{(k)}$ in Algorithm 2 indicates the k-level vertices of the EIG tree T_i. The notation TxRxMAC$(T_{i\to\mathcal{N}_i}^{(k)}, T_{\mathcal{N}_i\to i}^{(k)})$ indicates that, using the OMC, node i transmits $T_i^{(k)}$ to each node $j \in \mathcal{N}_i$, and receives $T_j^{(k)}$ from each node $j \in \mathcal{N}_i$.

We use UPDATE(T_i), to update the EIG tree T_i after the arrival of new information, and the procedure DECIDE(T_i) to infer the network topology based on the EIG tree. The n-stage EIG algorithm guarantees that if the subgraph of good nodes is connected, then each good node will decide on the same topological view.

Algorithm 2 The EIG Byzantine General's Algorithm

```
procedure EIGByzMAC(N_i)
  T_i^(0) := N_i
  for k = 1, ... n do
    while t ∈ S_{6+k} do
      TxRxMAC(T^(k)_{i→N_i}, T^(k)_{N_i→i})
      Update(T_i)
    end while
  end for
  Decide(T_i)
end procedure
```

C. The Consistency Check Phase

Unfortunately, a fundamental difficulty is that malicious nodes along a path $1, \dots, n$ may have generated false time stamps in the Neighbor Discovery Phase, and thus corrupted the measured relative skews between adjacent nodes. There may be several connecting paths infiltrated by bad nodes that thereby generate different values for the relative skew. It is impossible to determine the correct path from the relative skews alone. Every pair of such inconsistent paths corresponds to an inconsistent cycle in which the skew product is not equal to one. We use an algorithm called Consistency Check to identify the path that generated the correct relative skew.

Consistency Check works by circling a timing packet around every cycle in which the skew product differs from one by more than a desired maximum skew error ϵ_a. At its conclusion, the test removes at least one link with a malicious endpoint from the cycle, eliminating a connecting path. During the test, each node in such a cycle is obliged to append a receive time-stamp and a send time-stamp generated by the local clock before forwarding the packet to the next node. These time-stamps must satisfy a delay bound condition; the send and receive times cannot differ by more than 1 clock count. A node fails the consistency check otherwise, or if its time stamps do not agree with its declared relative skew. The key idea is that if the test starts after a sufficiently large amount of time has elapsed, the clock estimates based on faulty relative skews will have diverged so extensively from the actual clocks that at least one malicious node in the cycle will find it impossible to generate time-stamps that are consistent with its declared relative clock skew and satisfy the delay bound condition (all proofs are in Section VI):.

Theorem IV.1. *Let T_j be the start-time of the Consistency Check for the jth inconsistent cycle, consisting of nodes $i_1, \dots, i_m$. At least one malicious node in cycle j will violate a consistency check condition, if $T_j > \frac{\hat{a}_{i_m,i^*}(m+1)K+\epsilon_b}{\epsilon_a}$ where i^* is the node with the smallest skew product $\hat{a}_{i^*,i_1}$.*

Algorithm 3 depicts Consistency Check. Given a cycle j, the indices k and m denote nodes that follow and precede node i respectively in the cycle. If node i is the leader of the cycle, i..e., the node with smallest ID, then node i initiates the timing packet that traverses the cycle and transmits it to node k. Otherwise, node i waits for the timing packet to arrive from node m before forwarding it to node k.

Algorithm 3 Consistency Check Algorithm at Node i

```
procedure ConsistencyCheck
    START := ((n+1)(a_max)^(n+1) + (n+1)(a_max)^(n+1) U_0) / ε_a
    for each cycle C_j do
        k = Next(C_j)
        m = Prev(C_j)
        if i=Leader(C_j) and t ≥ START then
            Transmit(TIM_{i→k})
        else if i ∈ C_j then
            Receive(TIM_{m→i})
            Transmit(TIM_{i→k})
        end if
    end for
end procedure
```

After all inconsistent cycles have been tested, each node i disseminates the set of all timing packets $\mathcal{T}_i$ it received to other nodes. The EIG algorithm is used to ensure a common view of the timing packets generated. Each node removes from the topology any link whose endpoints generate time-stamps inconsistent with its declared relative skew or violated the delay bound. The complete phase is shown in Algorithm 4.

Algorithm 4 The Network Discovery Phase at Node i

```
procedure NetworkDiscovery
    EIGByzMAC(N_i)
    ConsistencyCheck
    EIGByzMAC(T_i)
end procedure
```

At the conclusion of Network Discovery Phase node i shares a common view of the network topology with all other good nodes. As a result, the network can designate the node with smallest ID as the *reference clock*. Furthermore, each node i has an estimate of the reference clock $\tau_i^r(t)$ with respect to its local clock t using the formula $\hat{\tau}_i^r(t) := \hat{a}_{ri}t$, where estimated $\hat{a}_{ri}$ and actual relative skews a_{ri} differ by at most ϵ_a.

D. The Scheduling Phase

In the Scheduling Phase the good nodes in the network obtain a common schedule governing the transmission and reception of data packets. A "schedule" is simply a sequence of CTVs, each with specified start and end times. Each node i divides the Data Transfer Phase into time-slots, and assigns a CTV to each time-slot so that the resulting throughput vector is utility optimal. All the good nodes independently arrive at the same schedule since they independently optimize the same utility function over the same $\mathcal{C}$ (ties broken lexicographically).

Since the good nodes must conform to a common schedule, each node i generates a local estimate of the reference clock $\hat{\tau}_i^r(t)$ with respect to its local clock t, as described in the Network Discovery Phase. However, this estimate may not be perfectly accurate; some of the nodes on a path along which relative skew is estimated may be malicious and can introduce an error of at most ϵ_a into the computed relative skew. To address this, the time-slots are separated by a dead-time of size D, where given any pair of nodes (i, j), D is chosen to satisfy $|\hat{\tau}_i^r(t) - \hat{\tau}_j^r(\tau_i^j(t))| \leq D$.

Finally, $n^2(n-1)$ time-slots are enough to guarantee that every pair of nodes can communicate once in either direction, via multihop routing, during Data Transfer Phase. The algorithm UtilityMaximization($\mathcal{C}$) for the Scheduling Phase is depicted in Algorithm 5. At the end of Scheduling Phase, node i shares a common utility maximizing schedule with other good nodes.

Algorithm 5 The Scheduling Phase at Node i

```
procedure Scheduling
    UtilityMaximization(C)
end procedure
```

E. The Data Transfer Phase

In this phase the nodes exchange data packets using the generated schedule. It is divided into time-slots, with each assigned a CTV, a rate vector, and set of packets for each transmitter in the set. To prevent collisions resulting from two nodes assigning themselves to different time slots due to timing error, node i begins transmission D time-units after the start of the time-slot. The transmitted packet is then guaranteed to arrive at the receiver in the same time slot, for appropriate choice of D and time-slot size B_{slot}.

Algorithm 7 defines this phase, with m_k denoting a message to be transmitted or received by node i in the k_{th} slot, T_{start} the start time of the phase measured by the local estimate of the reference clock $\hat{\tau}_i^r(t)$), $S_k = [t_k, t_{k+1}), k = 1, \ldots, N$ the time-slots of the phase with $N = n^2(n-1)$, $t_1 = T_{start}$, and $t_{k+1} := t_k + B_{slot} + 2D$, and $TX(k)$ and $RX(k)$ the CTV, and receiving nodes during slot k.

Algorithm 6 The Data Transfer Phase at Node i

```
procedure DataTransfer(T_start)
    for k=1,...,N do
        if t ∈ S_k and t ≥ t_k + D and i ∈ TX(k) then
            Transmit(m_k)
        else if t ∈ S_k and i ∈ RX(k) then
            Receive(m_k)
        end if
    end for
end procedure
```

F. The Verification Phase

However, malicious nodes may not cooperate in the Data Transfer Phase. So whenever a scheduled packet fails to arrive at node j, it adds the offending CTV and associated packet number to a list, and disseminates the list in the Verification Phase using the EIG Byzantine General's algorithm. These CTVs are then permanently further pruned from the collection of feasible CTVs. With L_k denoting the list that failed during the kth iteration of the Data Transfer Phase, the set $\mathcal{C}_k$ of feasible CTVs during the kth iteration of the Scheduling Phase is updated to $\mathcal{C}_{k+1} = \mathcal{C}_k \setminus L_k$ in Algorithm 7.

All communication can be scheduled into slots separated by a dead-time of $2D$. Within each of the n stages of the EIG

Byzantine General's algorithm, there are $n(n-1)$ pairs of nodes that may communicate, and at most n nodes on the connecting path. Therefore, the total number of time slots required is $n^3(n-1)$.

At the conclusion of the phase, the good nodes again share a common view of the set of feasible CTVs for the next iteration of the Scheduling Phase.

Algorithm 7 The Verification Phase at Node i

```
procedure VERIFICATION
    EIGBYZ(L_k)
    UPDATE(C_{k+1})
end procedure
```

G. The Steady State

The network cycles through Scheduling, Data Transfer, and Verification Phases for n_{iter} iterations. Eventually, by finiteness, it converges to a set of CTVs, and a utility-maximizing schedule over it. The overall protocol is in Algorithm 8.

Algorithm 8 The Complete Protocol

```
NEIGHBORDISCOVERY
NETWORKDISCOVERY
for k = 1, ..., n_iter do
    SCHEDULING(C_k)
    DATATRANSFER(t)
    VERIFICATION
end for
```

V. The Main Results

Our main result, elaborated on in Theorem VI.2, is:

Theorem V.1. *Consider a network that satisfies (N), (CO), (CL), (CR), (C) and (U). Given an arbitrary* ϵ*, where* $0 < \epsilon < 1$*, the protocol described above ensures that all the good nodes obtain a common estimate of the connected component that they are all members of, and achieves the utility* $(1-\epsilon) \min_{\{\alpha_k, \mathcal{D}_k \in \Delta\}} \max_{\{x \in \mathcal{C}(\mathcal{C}\setminus\mathcal{D}_k)\}} \sum_k \alpha_k U(x, F(\mathcal{C}\setminus\mathcal{D}_k))$*, where* α_k*, the fraction of the operating lifetime in which the concurrent transmission vectors in* $\mathcal{D}_k$ *are disabled, satisfies* $\sum_k \alpha_k = 1$.

Some important consequences are the following. Normally, one would expect that since the good nodes have to first declare their protocol and follow it, they can only attain "max-min," which is generally smaller than min-max. Since the latter can be attained (arbitrarily closely), it shows firstly that the bad nodes are unable to benefit from having a priori knowledge of the protocol. Second, since all that the bad nodes can benefit from is deciding which sets to disable, they are effectively limited to jamming and/or cooperating in each CTV. Other more Byzantine behaviors are not any more effective.

The example below shows why a bad node may prefer to "conform" rather than jam for some utility functions.

Example V.1. *Consider the network of Figure 1. Nodes 1 and 2 are good and in close proximity, while node 3 is bad and located far away. Consider the "fairness-based "utility function* $U(x) := \min\{x_{12}, x_{32}\}$*. If node 3 jams, then the connected component becomes* $\{1, 2\}$*, and the good nodes proceed to maximize only* x_{12}*, which node 3 can only slightly impinge because it is so far away from node 2. However, if node 3 cooperates, then the connected component is* $\{1, 2, 3\}$*, and the optimal solution for this "fair" utility function is to make* $x_{32} = x_{12}$*. However, link 32 being weak, it requires much more airtime than link 12, thus considerably reducing* x_{12}*.*

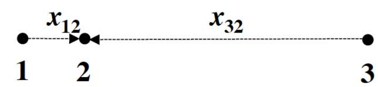

Fig. 1: Example V.1.

There are two important directions for future research that arise from just a perusal of this example. First, it shows that the choice of the utility function adopted in designing the protocol is important. As we have seen above the choice of the utility function $U(x) := \min\{x_{12}, x_{32}\}$ can be exploited by the bad nodes using a partial deafness attack (i.e., pretending to be very far away and thus receiving only a weak signal) to hurt the performance obtained by the good nodes. However, the utility function $U(x) := x_{12} + x_{32}$ is less exploitable by such a partial deafness attack. This points to the importance of properly choosing the utility function in the design of the protocol.

Second, as we have noted at the very outset of the paper, the results obtained are specifically with respect to the model of the capabilities available to the good and bad nodes. If nodes have availability to geographical positions of other nodes then claiming partial deafness due to being far away can be secured against. The points to the importance of investigating more general model classes that technology (e.g., GPS) can potentially support. Also, if malicious nodes have other means of attack, then that also would point to the need for generalizing the model.

The larger and important point is that we would have thereby elevated the theory of security from responding to attack by attack, to a theory that addresses models and utility functions. This is a much higher level playing field.

VI. Feasibility of Protocol and Optimality Proof

For the distributed wireless nodes to exchange data over the network, they must have a consistent view of a reference clock so that any activity will conform to this common schedule. For this, we consider the consistency check algorithm of Section IV-C. Consider a chain network $1, \ldots, n$, where the endpoints, nodes 1 and n are good, and the intermediate nodes $2, \ldots, n-1$ are bad. Note that this network can also be reduced to a cycle of size $n-1$ by making both endpoints the same node. We assume that the two good endpoints do not know if any of the intermediate nodes are bad.

Now suppose that each pair of adjacent nodes $(i, i-1)$ for $i = 2, \ldots, n$ has declared a set of relative skews and offsets $\{\hat{a}_{i,i-1}, \hat{b}_{i,i-1}\}$, and that each node in the chain knows this set.

The two good nodes wish to determine whether the declared skews are accurate, i.e., whether $a_{n,1} = \prod_{i=2}^{n} \hat{a}_{i,i-1}$. As per the consistency check, node 1 initiates a timing packet that traverses the chain from left to right after waiting a sufficiently long period of time. Each node in the chain is obligated to forward the packet after appending receive and time-stamps that satisfy the skew consistency and delay bound conditions.

In order to defeat this test, the bad nodes, having collectively declared a false set of relative skews and offsets, must support two sets of clocks for each node $i \in \{2, \dots, n\}$: a "left" clock $\tau^{i,l}(t)$ to generate receive time-stamps, and a "right" clock $\tau^{i,r}(t)$ to generate send-time stamps. The bad nodes are free to jointly select any set of clocks $\{\tau^{i,l}(t), \tau^{i,r}(t), \forall i = 2, \dots, n-1\}$ that are arbitrary functions of t, a much larger set than the affine clocks being emulated. However, we will show that if node 1 waits sufficiently long enough, there is no set of clocks $\{\tau^{i,l}(t), \tau^{i,r}(t), i = 2, \dots, n-1\}$ that can generate time-stamps which satisfy both conditions of the consistency check.

Let $r_{i,i-1}$ and $s_{i,i+1}$ denote the receive and send time-stamps generated by a bad node i with respect to the left and right clocks $\tau^{i,l}(t)$ and $\tau^{i,r}(t)$ respectively. Let $t_{i,l}$ and $t_{i,r}$ denote the time with respect to the global reference clock at which the receive and send time-stamps are generated at node i. We have $r_{i-1,i} := \tau^{i,l}(t_{i,l})$ and $s_{i,i+1} := \tau^{i,r}(t_{i,r})$. Let t_1 and t_n denote the time with respect to the global reference clock at which the timing packet was transmitted by node 1 and received by node n respectively. We have $s_{1,2} := \tau^1(t_1), r_{n-1,n} := \tau^n(t_n)$. To simplify notation we will define left and right clocks at the endpoints so that $t_{1,r} := t_1, t_{n,l} := t_n$ and $\tau^{1,r}(t_{1,r}) := \tau^1(t_1), \tau^{n,l}(t_{n,l}) := \tau^n(t_n)$.

In order to prove that both conditions of the consistency check cannot be satisifed by any set of clocks $\{\tau^{i,l}(t), \tau^{i,r}(t), i = 2, \dots, n-1\}$, we will assume that the first condition is satisfied, and show that second must fail. Therefore, the clocks must satisfy:

$$\tau^{i,l}(t_{i,l}) = a_{i,i-1}\tau^{i-1,r}(t_{i-1,r}) + b_{i,i-1} \text{ for } i \leq 2 \leq n. \quad (1)$$

In addition, by virtue of causality, we also have:

$$\tau^{i,l}(t_{i,l}) \leq \tau^{i,r}(t_{i,r}). \quad (2)$$

We prove that delay bound condition must be violated if node 1 waits for a sufficiently large period of time before before initiating the timing packet, i.e., if $\tau^1(t_1)$ is sufficiently large, then for some i, we have $\tau^{i,r}(t_{i,r}) - \tau^{i,l}(t_{i,l}) > K$. More precisely, we show $\sum_{i=2}^{n-1} \left(\tau^{i,r}(t_{i,r}) - \tau^{i,l}(t_{i,l})\right) > nK$, which implies that some node has violated delay bound condition. We have the following equality $\tau^{n,l}(t_{n,l}) = \tau^{1,r}(t_{1,r}) + S_1 + S_2$, where $S_1 := \sum_{i=2}^{n} \left(\tau^{i,l}(t_{i,l}) - \tau^{i-1,r}(t_{i-1,r})\right), S_2 := \sum_{i=2}^{n-1} \left(\tau^{i,l}(t_{i,l}) - \tau^{i-1,r}(t_{i-1,r})\right)$. The value S_2 is the sum of the forwarding delays. We will use (1) and (2) to obtain an upper bound on S_1. Inserting this upper bound and using the fact that $\tau^{n,l}(t)$ and $\tau^{1,r}(t)$ are both affine functions of t, will allow us to obtain a lower bound on S_2. The proof will then follow easily. We now obtain an upper bound on S_1 when the forward skew product $\prod_{i=2}^{j} \hat{a}_{i,i-1} \geq 1$ for all $j \geq 2$.

Lemma VI.1. *Suppose* $\prod_{i=2}^{j} a_{i,i-1} \geq 1$ *for* $2 \leq i \leq n$. *Then* $\sum_{i=2}^{n}(\tau^{i,l}(t_{i,l}) - \tau^{i-1,r}(t_{i-1,r})) \leq \left(\frac{\hat{a}_{n,1}-1}{\hat{a}_{n,1}}\right)\tau^{n,l}(t_{n,l})\sum_{i=2}^{n} \frac{\hat{b}_{i,i-1}}{\hat{a}_{i,1}}$.

Proof. We have by definition $\tau^{n+1,l}(t_{n,l}) := \hat{a}_{n+1,n}\tau^{n,r}(t_{n,r}) + \hat{b}_{n+1,n}$. For $n = 2$, we have $\tau^{2,l}(t_{2,l}) - \tau^{1,r}(t_{1,r}) = \left(\frac{a_{2,1}-1}{a_{2,1}}\right)\tau^{2,l}(t_{2,l}) + \frac{b_{2,1}}{a_{2,1}}$. Now assume the lemma is true for n. We will show that it also holds for $n+1$: $\sum_{i=2}^{n+1}(\tau^{i,l}(t_{i,l}) - \tau^{i-1,r}(t_{i-1,r})) = \sum_{i=2}^{n}(\tau^{i,l}(t_{i,l}) - \tau^{i-1,r}(t_{i-1,r})) + \tau^{n+1,l}(t_{n+1,l}) - \tau^{n,r}(t_{n,r}) \leq \left(\frac{\hat{a}_{n,1}-1}{\hat{a}_{n,1}}\right)\tau^{n,l}(t_{n,l}) + \sum_{i=2}^{n}\frac{\hat{b}_{i,i-1}}{\hat{a}_{i,1}} + \tau^{n+1,l}(t_{n+1,l}) - \tau^{n,r}(t_{n,r}) \leq \left(\frac{\hat{a}_{n,1}-1}{\hat{a}_{n,1}}\right)\tau^{n,r}(t_{n,r}) + \sum_{i=2}^{n}\frac{\hat{b}_{i,i-1}}{\hat{a}_{i,1}} + \tau^{n+1,l}(t_{n+1,l}) - \tau^{n,r}(t_{n,r}) = \left(\frac{\hat{a}_{n,1}-1}{\hat{a}_{n,1}}\right)\left(\frac{\tau^{n+1,l}(t_{n+1,l})-\hat{b}_{n+1,n}}{\hat{a}_{n+1,n}}\right) + \sum_{i=2}^{n}\frac{\hat{b}_{i,i-1}}{\hat{a}_{i,1}} + \left(\frac{\hat{a}_{n+1,n}-1}{\hat{a}_{n+1,n}}\right)\hat{\tau}^{n+1,l}(t_{n+1,l}) + \frac{\hat{b}_{n+1,n}}{\hat{a}_{n+1,n}} = \left(\frac{\hat{a}_{n+1,1}-1}{\hat{a}_{n+1,1}}\right)\tau^{n+1,l}(t_{n+1,l}) + \sum_{i=2}^{n+1}\frac{\hat{b}_{i,i-1}}{\hat{a}_{i,1}}$, which follow from the induction hypothesis above in the lemma statement, and the fact that $\tau^{n,r}(t_{n,r}) \geq \tau^{n,l}(t_{n,l})$ and $a_{i,1} \geq 1$ for all $2 \leq i \leq n+1$ (that is, the coefficient $\hat{a}_{i,1} - 1$ is negative). ☐

We next bound S_1 in the special case when the reverse skew product $\prod_{i=1}^{j} \hat{a}_{n-(i-1),n-i} \leq 1$ for all $j \geq 1$.

Lemma VI.2. *Suppose* $\prod_{i=1}^{j} a_{n-(i-1),n-i} \leq 1$ *for* $2 \leq j \leq n-1$. *Then* $\sum_{i=1}^{j}(\tau^{n-(i-1),l}(t_{n-(i-1),l}) - \tau^{n-i,r}(t_{n-i,r})) \leq (\hat{a}_{n,n-j} - 1)\tau^{n-j,r}(t_{n-j,r}) + \hat{b}_{n,n-1} + \sum_{i=n-j+1}^{n-1}\hat{a}_{n,i}\hat{b}_{i,i-1}$.

Proof. We have by definition $\tau^{n-(k-1),l}(t_{n-(k-1),l}) := \hat{a}_{n-(k-1),n-k}\tau^{n-k,r}(t_{n-k,r}) + \hat{b}_{n-(k-1),n-k}$. For $j = 1$, $\tau^{n,l}(t_{n,l}) - \tau^{n-1,r}(t_{n-1,r}) = (a_{n,n-1} - 1)\tau^{n-1,r}(t_{n-1,r})$.

Now assume the lemma holds for j. We will show that it must hold for $j+1$: $\sum_{k=1}^{j+1}(\tau^{n-(k-1),l}(t_{n-(k-1),l}) - \tau^{n-k,r}(t_{n-k,r})) = \sum_{k=1}^{j}(\tau^{n-(k-1),l}(t_{n-(k-1),l}) - \tau^{n-k,r}(t_{n-k,r})) + \tau^{n-j,l}(t_{n-j,l}) - \tau^{n-(j+1),r}(t_{n-(j+1),r}) \leq (\hat{a}_{n,n-j}-1)\tau^{n-j,r}(t_{n-j,r}) + \hat{b}_{n,n-1} + \sum_{k=n-j+1}^{n-1}\hat{a}_{n,k}\hat{b}_{k,k-1} + \tau^{n-j,l}(t_{n-j,l}) - \tau^{n-(j+1),r}(t_{n-(j+1),r}) \leq (\hat{a}_{n,n-j} - 1)\tau^{n-j,l}(t_{n-j,l}) + \hat{b}_{n,n-1} + \sum_{k=n-j+1}^{n-1}\hat{a}_{n,k}\hat{b}_{k,k-1} + \tau^{n-j,l}(t_{n-j,l}) - \tau^{n-(j+1),r}(t_{n-(j+1),r}) \leq (\hat{a}_{n,n-(j+1)} - 1)\tau^{n-(j+1),r}(t_{n-(j+1),r}) + \hat{b}_{n,n-1} + \sum_{k=n-j}^{n-1}\hat{a}_{n,k}\hat{b}_{k,k-1}$. The above follow from induction hypothesis in Lemma VI.2, since $\tau^{i,l}(t_{i,l}) \leq \tau^{i,r}(t_{i,r})$ and $\hat{a}_{n,n-j} \leq 1$ for $1 \leq j \leq n-1$ (that is, the coefficient $\hat{a}_{n,n-j} - 1$ is negative), and from substitution into $\tau^{n-j,l}(t_{n-j,l})$ and simplification. ☐

We will combine both special cases in Lemma VI.1 and Lemma VI.2 to obtain an upper bound on S_1. First we define i^* as the node with the smallest skew product $\hat{a}_{i^*,1}$ in the chain network, that is less than one. That is, $\hat{a}_{i^*,1} = \min_k \hat{a}_{k,1}$ and $\hat{a}_{i^*,1} \leq 1$. If no such node exists, set $i^* = 1$.

Now we consider an arbitrary set of skews $\{\hat{a}_{i,i-1}, i = 2, \dots, n\}$. Next we show that if $i^* \geq 2$ then the forward skew product starting from i^* is greater than 1, and the reverse skew product starting from $i^* - 1$ is always less than one.

Lemma VI.3. *If* $i^* \geq 2$ *then* $\hat{a}_{j,i^*} \geq 1$ *for* $i^* + 1 \leq j \leq n$ *and* $\hat{a}_{i^*,i^*-k+1} \leq 1$ *for* $1 \leq k \leq i^*$. *Otherwise,* $\hat{a}_{j,1} \geq 1$ *for* $2 \leq j \leq n$.

Proof. Consider $i^* \geq 2$, and suppose the first part of the assertion is false. I.e., for some j', $\hat{a}_{j'i^*} < 1$. It follows that $\hat{a}_{j'1} = \hat{a}_{j'i^*}\hat{a}_{i^*1} \leq \hat{a}_{i^*1}$. But then j' is a node with a smaller skew product $\hat{a}_{j1}$ than node i^*, which contradicts the definition of i^*. Now suppose that the second part of the assertion is false. I.e., for some j' we have $\hat{a}_{i^*j'} > 1$. It follows that $\hat{a}_{i^*1} = \hat{a}_{i^*j'}\hat{a}_{j'1} \geq \hat{a}_{j'1}$. But then j' is a node with a smaller skew product than node i^*, which again contradicts the definition of i^*. Now consider the case when $i^* = 1$. Then by definition of i^* it follows that $\hat{a}_{j1} \geq 1$ for all $2 \leq j \leq n$. □

We now obtain an upper bound on S_1 for arbitrary skews.

Lemma VI.4. *Suppose $i^* \geq 2$. We have the following inequality: $\sum_{j=2}^{n} \tau^{j,l}(t_{j,l}) - \tau^{j-1,r}(t_{j-1,r}) \leq (\hat{a}_{i^*,1} - 1)\tau^{1,r}(t_{1,r}) + \left(\frac{\hat{a}_{n,i^*}-1}{\hat{a}_{n,i^*}}\right)\tau^{n,l}(t_{n,l}) + \frac{\hat{b}_{n,1}}{\hat{a}_{n,i^*}}$.*

Proof. $\sum_{j=2}^{n} \tau^{j,l}(t_{j,l}) - \tau^{j-1,r}(t_{j-1,r}) = \sum_{j=2}^{i^*} \tau^{j,l}(t_{j,l}) - \tau^{j-1,r}(t_{j-1,r}) + \sum_{j=i^*+1}^{n} \tau^{j,l}(t_{j,l}) - \tau^{j-1,r}(t_{j-1,r}) = (\hat{a}_{i^*,1} - 1)\tau^{1,r}(t_{1,r}) + \left(\frac{\hat{a}_{n,i^*}-1}{\hat{a}_{n,i^*}}\right)\tau^{n,l}(t_{n,l}) + \frac{\hat{b}_{n,1}}{\hat{a}_{n,i^*}}$.which follow by applying Lemma VI.2 and Lemma VI.1, by multiplying the terms in each summation by $\frac{\hat{a}_{n,i^*}}{\hat{a}_{n,i^*}}$ and simplifying, and from the definitions of $\hat{b}_{ij}$ and $\hat{d}_{ji}^{(i)}$. □

Now that we have an upper bound on S_1, we can obtain a lower bound on S_2, the sum of the forwarding delays.

Lemma VI.5. *The sum of forwarding delays in the chain network satisfies: $\sum_{j=2}^{n-1} \left(\tau^{j,l}(t_{j,l}) - \tau^{j,r}(t_{j,r})\right) \geq \frac{(a_{n,1}-\hat{a}_{n,1})}{\hat{a}_{n,i^*}}\tau^{1,r}(t_{1,r}) + \frac{(b_{n,1}-\hat{b}_{n,1})}{\hat{a}_{n,i^*}}$.*

Proof. $\sum_{j=2}^{n-1} \left(\tau^{j,l}(t_{j,l}) - \tau^{j,r}(t_{j,r})\right) = \tau^{n,l}(t_{n,l}) - \tau^{n,r}(t_{n,r}) - \sum_{j=2}^{n} \tau^{j,l}(t_{j,l}) - \tau^{j-1,r}(t_{j-1,r}) \geq \tau^{n,l}(t_{n,l}) - \tau^{n,r}(t_{n,r}) - (\hat{a}_{i^*,1} - 1)\tau^{1,r}(t_{1,r}) - \left(\frac{\hat{a}_{n,i^*}-1}{\hat{a}_{n,i^*}}\right)\tau^{n,l}(t_{n,l}) - \frac{\hat{b}_{n,1}}{\hat{a}_{n,i^*}} \geq \frac{\tau^{n,l}(t_{1,r})}{\hat{a}_{n,i^*}} - \hat{a}_{i^*,1}\tau^{1,r}(t_{1,r}) - \frac{\hat{b}_{n,1}}{\hat{a}_{n,i^*}} = \frac{(a_{n,1}-\hat{a}_{n,1})}{\hat{a}_{n,i^*}}\tau^{1,r}(t_{1,r}) + \frac{(b_{n,1}-\hat{b}_{n,1})}{\hat{a}_{n,i^*}}$, which follow by applying Lemma VI.4, because $t_{n,l} \geq t_{1,r}$ since node n could not have received the timing packet before node 1 transmitted it, and since node n's clock is relatively affine with respect to 1's. □

To complete the proof, we show that if the start time of the consistency check is sufficiently large, and the left and right clocks $\{\tau^{i,l}(t_{i,l}), \tau^{i,r}(t_{i,r})\}$ satisfy the parameter consistency condition, then at least one node will violate delay bound condition.

Proof. We assume node 1 is a good node. Now $\frac{(a_{n,1}-\hat{a}_{n,1})}{\hat{a}_{n,i^*}}\tau^{1,r}(t_{1,r}) + \frac{(b_{n,1}-\hat{b}_{n,1})}{\hat{a}_{n,i^*}} > nK$. But by Lemma VI.5 the LHS of this inequality is the lower bound of the sum of the delays in the chain $\sum_{j=2}^{n} \left(\tau^{j,l}(t_{j,l}) - \tau^{j,r}(t_{j,r})\right)$. By substitution, $\sum_{j=2}^{n} \left(\tau^{j,l}(t_{j,l}) - \tau^{j,r}(t_{j,r})\right) > nK$. It follows that for some malicious node $j \in \{2, \ldots, n\}$, $\tau^{j,l}(t_{j,l}) - \tau^{j,r}(t_{j,r}) > K$ which violates the delay bound condition. □

Now we can show that neighbor and network discovery phases together allow the good nodes to form a rudimentary network, where the good nodes have the same topological view and consistent estimates of a reference clock. The first obstacle is that the protocol is composed of stages that must be completed sequentially by all the nodes in the network, even prior to clock synchronization. Suppose that $[t_k, t_{k+1})$ is the interval allocated to the kth stage. Any messages transmitted between adjacent good nodes must arrive in the same interval they were transmitted. Since send-times are measured with respect to the source clock, and receive-times with respect to the destination clock, the intervals must be chosen large enough to compensate for the maximum clock divergence caused by skew $a_{ij} \leq a_{max}$ and offset $b_{ij} \leq a_{max}U_0$.

Lemma VI.6. *There exists a sequence of adjacent time-intervals $[t_k, t_{k+1})$ and corresponding schedule that guarantees any message of size W transmitted (via OMC) by node i in the interval $[t_k, t_{k+1})$ (as measured by i's clock) will be received by node j in the same interval as measured by node j's clock.*

Proof. Set $t_{k+1} := (a_{max})^2 t_k + 2(a_{max})^3 U_0 + (a_{max})^3 T_{MAC}(W)$. Suppose a message from node i to node j during $[t_k, t_{k+1})$ is transmitted (via the OMC) at $t_s := a_{max}t_k + (a_{max})^2 U_0$ with respect to node i's clock. By substitution and simplification it follows that $\tau_i^j(t_s) \geq t_k$ and $\tau_i^j(t_s + T_{MAC}(W)) < t_{k+1}$. Hence $\tau_i^j([t_s, t_s + T_{MAC}(W))) \subset [t_k, t_{k+1})$, and so j receives this message during the same interval with respect to j's clock. □

Theorem VI.1. *After Network Discovery, the good nodes have a common view of the topology and consistent estimates (to within ϵ_a) of the skew of the reference clock.*

Proof. From Lemma VI.6 all good nodes will proceed through each stage of Neighbor and Network Discovery Phases together, and therefore establish link certificates with their good neighbors. Since they form a connected component, the good nodes obtain a common view of their link certificates using the EIGByzMAC algorithm and the schedule in Lemma VI.6. The good nodes can therefore infer the network topology and the relative skews of all adjacent nodes based upon the collection of link certificates. Using Consistency Check, the good nodes can eliminate paths along which bad nodes have provided false skew data. The good nodes can disseminate this information to each other using the EIGByzMAC algorithm and Lemma VI.6 and thus obtain consistent estimates of the reference clock to within ϵ_a. □

Lemma VI.7. *The sequence of adjacent intervals $[t_j, t_{j+1})$, $j = 0, \ldots, k$ is contained in $[t_0, c_1 t_0 + c_2 W)$ where constants c_1 and c_2 depend on a_{max}, k, U_0, and n.*

Proof. For the OMC $T_{MAC}(W) \leq cW$, where c depends on a_{max}, and n. The result for $k = 1$ follows from definition of t_k, and substitution of cW into $T_{MAC}(W)$, and for general k by induction and definition of t_k. □

Lemma VI.8. *The time to complete Neighbor and Network Discovery Phases $T_{nei} + T_{net}$ is less than $c_1 \log T_{life} + \frac{c_2}{\epsilon_a}$ where c_1, c_2 depend only on n, a_{max}, U_0.*

Proof. From Algorithms 1, 2, 3 and 4 there are at most $6 + n + n|C| + n$ protocol stages in the Neighbor and Network Discovery Phases. Hence the time required is at most $c_1 t_0 + c_2 W$, where W is the size of a message to be transmitted, and c_1, c_2 are constants depending on the number of protocol stages a_{max}, U_0, n. The maximum size of a message is proportional to the timing packet size $\log T_{life}$. To account for the effect of the minimum start-time T_s for the consistency check, we can assume the worst case that the T_s comes into effect during the first protocol stage (instead of later in the Network Discovery Phase). From Theorem IV.1 the consistency check start-time is at most $\frac{c}{\epsilon_a}$, where c depends on U_0, a_{max}, n. Substitution into t_0 proves the lemma. □

Lemma VI.9. *The time required for the Data Transfer Phase is at most $c_3 B + c_4 D$ where B is the time spent transmitting data packets, D is the size of the dead-time separating time slots, and c_3, c_4 depend on n alone.*

Proof. The total number of time-slots for data transfer between all source-destination pairs is $n^2(n-1)$, each supporting data transfer of size B_s and a dead-time D. □

Lemma VI.10. *The time required for the Verification Phase is at most $c_5 D$ where c_5 depends on n alone.*

Proof. In each stage of the EIG Byzantine General's algorithm, there are at most $n!$ vertex values that must be transmitted with each node in the neighborhood. The value of a vertex is a list of CTVs. There are at most 2^n CTVs and at most n nodes in a CTV. Therefore the size of any message to be transmitted by a node during EIG algorithm is at most cD, where c is a constant dependent on n. Since there are $n(n-1)$ possible source-destination pairs, there are at most $n(n-1)$ time slots in each stage, separated at the beginning and end by a dead-time D. Therefore the duration of each stage is at most $cD + n(n-1)2D$. There are at most n stages. □

Theorem VI.2. *The protocol ensures that the network proceeds from startup to a functioning network carrying data. There exists a selection of parameters n_{iter}, D, B, ϵ_a and T_{life} that achieves min-max utility over the allowed set, to within a factor ϵ, where the min is over all policies of the bad nodes that can only adopt two actions in each CTV: conform to the protocol and/or jam. The achieved utility is ϵ-optimal.*

Proof. We begin by choosing parameters so that the protocol overhead, which includes Neighbor Discovery, Network Discovery, Verification, all dead-times, and iterations converging to the final rate vector, is an arbitrarily small fraction of the total operating lifetime. With $\hat{\tau}_i^r(t) := \hat{a}_{ri} t$ the estimate of reference clock r with respect to the local clock at node i, the maximum difference in nodal estimates is bounded as $|\hat{\tau}_i^r(\tau^i(t)) - \hat{\tau}_k^r(\tau_i^k(\tau^i(t)))| \leq 2(a_{max})^2 \epsilon_a T_{life} + (a_{max})^2 U_0$. With k_r be the number of rate vectors in the rate region, we can choose n_{iter}, D, B, ϵ_a and T_{life} to satisfy: $\frac{n_{iter}}{n_{iter} + 2^n k_r} \geq 1 - \epsilon_l$, $\frac{B}{c_1 \log T_{life} + \frac{c_2}{\epsilon_a} + B + c_3 D + c_4 D} \geq 1 - \epsilon_d$, $n_{iter}((c_1 \log T_{life} + \frac{c_2}{\epsilon_a} + B + c_3 D + c_4 D) \leq T_{life}$, $2(a_{max})^2 \epsilon_a T_{life} + (a_{max})^2 U_0 \leq D$. These ensure that the rate loss due to failed CTVs is arbitrarily small, the time spent transmitting data is an arbitrarily large fraction of the duration of that iteration, the operating lifetime is large enough to support n_{iter} protocol iterations, and the dead-time D is large enough to tolerate the maximum divergence in clock estimates caused by skew error ϵ_a.

Consider a protocol iteration in which the estimate of the set of feasible non-disabled concurrent transmission vectors is correct and equal to $\mathcal{D}_k$ for some k. Suppose x achieves the maximum utility for $\mathcal{D}_k$ over the nodes in the same component as the good nodes. No protocol can do better when $\mathcal{D}_k$ is disabled. The proposed protocol attains $x(1 - \epsilon_d)$ during this iteration. By design, the fraction of the iterations in which this estimate is correct is $(1 - \epsilon_l)$. The theorem follows by noting that the fraction of the operating lifetime in which $\mathcal{D}_k$ is disabled is α_k. □

VII. Concluding Remarks

We have presented a complete suite of protocols that enables a collection of good nodes interspersed with bad nodes to form a functioning network from start-up, operating at a utility-optimal rate vector, regardless of what the bad nodes conspire to do, under a certain system model. Further, the attackers cannot decrease the utility any more than they could by just conforming to the protocol or jamming on each CTV.

This paper is only an initial attempt to develop a theoretical foundation for a much needed holistic all-layer approach to secure wireless networking, and there are several open issues, some more addressable than others. This paper essentially assumes the network to be time-varying; in particular that it is not mobile. Extension to quasi-static mobility appears feasible, and is worth doing. Since the protocol presented has poor transient behavior, though overall asymptotically optimal, it is worthwhile to explore how to increase efficiency in the transient phase. One possibility is to use mere flooding algorithms and not invoke Byzantine agreement unless it turns out that there are indeed malicious nodes, which fact can be checked in a post-facto verification phase. An important potential generalization is to allow probabilistic communication. Showing similar results in a "with high probability" sense appears feasible. Finally, the gap between information theory and security appears formidable, since the former does not recognize complexity-based cryptography.

Much further work remains to be done.

References

[1] A. Bar-Noy, D. Dolev, C. Dwork, and H. R. Strong. Shifting gears: changing algorithms on the fly to expedite Byzantine agreement. PODC '87, pages 42–51, New York, NY, USA, 1987. ACM.

[2] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. Macaw: a media access protocol for wireless lan's. In *ACM SIGCOMM Computer Communication Review*, volume 24, pages 212–225. ACM, 1994.

[3] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash. A feedback-based scheme for improving TCP performance in ad hoc wireless networks. *IEEE Personal Communications Magazine*, 8(1):34–39, Feb. 2001.

[4] J. Choi, J. T. Chiang, D. Kim, and Y.-C. Hu. Partial deafness: A novel denial-of-service attack in 802.11 networks. In *Security and Privacy in Communication Networks*, volume 50 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 235–252. Springer Berlin Heidelberg, 2010.

[5] T. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR). RFC 3626, Oct. 2003.

[6] Nikolaos M. Freris, Scott R. Graham and P. R. Kumar, Fundamental Limits on Synchronizing Clocks over Networks. *IEEE Transactions on Automatic Control*, 56:1352–1364, 2011.

[7] Z. Fu, B. Greenstein, X. Meng, and S. Lu. Design and implementation of a tcp-friendly transport protocol for ad hoc wireless networks. In *IEEE International Conference on Network Protocols'02*, 2002.

[8] Y.-C. Hu, A. Perrig, and D. Johnson. Packet leashes: a defense against wormhole attacks in wireless networks. In *INFOCOM 2003*, volume 3, pages 1976 – 1986 vol.3, march-3 april 2003.

[9] Y.-C. Hu, A. Perrig, and D. B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. WiSec '03, pages 30–40, 2003.

[10] Y.-C. Hu, A. Perrig, and D. B. Johnson. Ariadne: a secure on-demand routing protocol for ad hoc networks. *Wirel. Netw.*, 11(1-2):21–38, Jan. 2005.

[11] IEEE Protocol 802.11. Draft standard for wireless lan: Medium access control (MAC) and physical layer (PHY) specifications. IEEE, July 1996.

[12] D. Johnson, Y. Hu, and D. Maltz. The dynamic source routing protocol (dsr) for mobile ad hoc networks for ipv4. RFC 4728, Feb. 2007.

[13] V. Kawadia and P. R. Kumar. Principles and protocols for power control in wireless ad-hoc networks. *IEEE Journal on Selected Areas in Communications*, 23:76–88, 2005.

[14] J. Liu and S. Singh. ATCP: TCP for mobile ad hoc networks. *IEEE J-SAC*, 19(7):1300–1315, July 2001.

[15] S. Marti, T. J. Giuli, K. Lai, M. Baker, et al. Mitigating routing misbehavior in mobile ad hoc networks. In *International Conference on Mobile Computing and Networking: Proceedings of the 6 th annual international conference on Mobile computing and networking*, volume 6, pages 255–265, 2000.

[16] J. P. Monks, V. Bharghavan, and W.-M. Hwu. A power controlled multiple access protocol for wireless packet networks. In *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 219–228. IEEE, 2001.

[17] M. Pease, R. Shostak and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27:228234, 1980.

[18] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on demand distance vector (aodv) routing. RFC 3561, July 2003.

[19] C. E. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing. In *SIGCOMM*, pages 234–244, London, UK, Aug. 1994.

[20] J. Ponniah, Y.-C. Hu, and P. R. Kumar. An orthogonal multiple access coding scheme. *Communications in Information and Systems*, 12:41–76, 2012.

[21] M. Poturalski, P. Papadimitratos, and J.-P. Hubaux. Secure neighbor discovery in wireless networks: formal investigation of possibility. ASIACCS '08, pages 189–200, New York, NY, USA, 2008. ACM.

[22] W. R. Stevens and G. R. Wright. *TCP/IP Illustrated: Vol. 2: The Implementation*, volume 2. Addison-Wesley Professional, 1995.

[23] F. Wang and Y. Zhang. Improving TCP performance over mobile ad-hoc networks with out-of-order detection and response. In *Proceedings of the third ACM international symposium on Mobile ad hoc networking and computing*, pages 217–225. ACM Press, 2002.