A Clean Slate Design for Secure Wireless Ad-Hoc Networks - Part 1: Closed Synchronized Networks

Jonathan Ponniah
Department of Electrical and Computer Engineering
Texas A&M

Yih-Chun Hu
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

P. R Kumar
Department of Electrical and Computer Engineering
Texas A&M

***Abstract*—We propose a clean-slate, holistic approach to the design of secure protocols for wireless ad-hoc networks. We design a protocol that enables a collection of distributed nodes to emerge from a primordial birth and form a functioning network. We consider the case when nodes are synchronized and the network is closed, in that no other nodes can join. We define a game between protocols and adversarial nodes, and describe a protocol that is guaranteed to achieve the max-min payoff regardless of what the adversarial nodes do. Moreover, even though the adversarial nodes always know the protocol a priori, we show an even stronger result; the protocol is guaranteed to achieve the min-max payoff. Hence there is a saddle point in the game between protocols and adversarial strategies. Finally, we show that the adversarial nodes are in effect, strategically confined to either jamming or conforming to the protocol. These guarantees are contingent on a set of underlying model assumptions, and cease to be valid if the assumptions are violated.**

I. Introduction

Wireless *ad-hoc* networks, the focus of this paper, are wireless networks without the defining feature of a centralized controller. Rather, the nodes themselves are responsible for determining in real-time, the routes over which packets are transmitted, the priorities assigned to each data flow, a common schedule of transmissions and receptions to realize these priorities, and the corresponding transmit power levels, modulation schemes, and encoding schemes. Due to the multiplicity of tasks that are required to maintain a functioning network, every facet of operation is usually carried out by a specially constructed suite of protocols.

The problem we consider in this paper, concerns the design of *secure* protocols for wireless ad-hoc networks; protocols that function even when a subset of the nodes in the network are committed to breaking the network operation by any possible means, including cynical exploitation of the very protocol they are expected to follow.

This material is based upon work partially supported by NSF Contract CNS-1302182, AFOSR Contract FA9550-13-1-0008, and NSF Science & Technology Center Grant CCF-0939370.

The design of secure complex systems in general is a challenging task. One common approach, defense in depth, [1], attempts to identify the "attack surface," all the different possible points of entry through which an attacker can breach the system, and erect countermeasures for each of these attacks. However, in complex systems, the vulnerabilities and weakness in a design may not be immediately obvious or evident, and there is always a possibility of some vulnerability or gap escaping detection. Often, so-called secure protocols are the outcome of a predictable sequence of events: first the protocol is introduced into service, next, a vulnerability is discovered post-facto, and finally, a fix or patch is devised to plug the security gap [2]. This approach amounts to an arms race between the protocols and adversarial nodes in which each subsequent fix or patch is countered with an even more sophisticated attack. At no point in this process is it possible to determine whether the last of the vulnerabilities has been found or if there are yet more to be discovered.

In this paper we describe an alternative clean-slate, holistic, model-based approach to security with provable security and performance guarantees. Our approach results in a complete protocol suite with the following hard security guarantee:

(G1) The system is secure against all possible attacks in the model of the wireless environment and system.

In other words, the system is guaranteed to retain functionality regardless of what the adversarial nodes do, where "functionality" implies an ability to transfer data between legitimate source destination pairs. To make this statement more quantitative, we need to define a measure of the system functionality and describe how far the adversarial nodes can displace the network from the optimal measure. To that end, suppose that there is a utility function $U(x)$ of the network throughput vector x between all source destination pairs. Now, we can have a more formal definition of security: the wireless ad-hoc network system is able to closely operate at a utility-optimal throughput, even when the internal processes,

978-3-9018-8273-9/15/ ©2015 IFIP

in this case adversarial nodes, behave arbitrarily.

We define a game between protocols and adversarial strategies. First, the protocol p is chosen and revealed to all the nodes in the network. Then the adversarial nodes choose a strategy q_p in response to the protocol. The legitimate and adversarial nodes execute (p, q_p) for the entire operating lifetime, resulting in an effective throughput vector x. The payoff $J(p, q_p)$ to the game depends on some function of U and x (as we will discuss, the payoff might be different from $U(x)$ if the protocol is able to discern the identities of the adversarial nodes).

We will describe a protocol p^* that is optimal in the following sense:

(G2) The protocol p^* is near max-min optimal with respect to the pay-off function J, regardless of the strategy of the adversarial nodes q_p.

The payoff in (G2) would appear to be the best any protocol could achieve; in the game, the adversarial strategy is always selected after the protocol. However, we will show that the protocol achieves an even better payoff.

(G3) The protocol p^* is near min-max optimal, regardless of the strategy of the adversarial nodes q.

The claim (G3) implies there is a saddle-point in the game between protocols and adversarial strategies; the adversarial nodes gain no advantage in terms of the payoff, from knowing the protocol a priori. We will show an even stronger result:

(G4) The protocol p^* is near min-max optimal, where the minimization is limited to those adversarial strategies in which concurrent transmission vectors are actively disabled.

We will expound upon (G4) in greater detail, but essentially this claim implies that the adversarial nodes are confined to either jamming or conforming to the protocol; both actions which when consistently applied, cannot be prevented. The claims in (G2), (G3), and (G4) are all versions of the central claim (G1), and individually offer a comprehensive guarantee of security against all possible attacks, without the usual condition that the attacks be identified a priori. Instead, (G1)-(G4) imply that the network can operate securely without even knowing the full scope of the strategies available to the adversarial nodes.

However, these guarantees are contingent on some underlying premises, or what we refer to as the model assumptions. The guarantees cease to be valid if the model assumptions are violated. The fundamental difference between the approach proposed in this paper and defense-in-depth, is in the idea that the measure of the protocol security lies in the robustness of the model assumptions not in the number of layers protecting narrowly defined operations. Ultimately, we argue that this approach is a more logically sound and effective way of providing and evaluating system security than an arm's race.

This framework and the claims (G1)-(G4) were first proposed in [3] for a more general communication model with unsynchronized nodes. In this paper, we introduce a simpler model with synchronized nodes, so that the issues not discussed in the previous work can receive a more thorough treatment.

II. The Problem Formulation and the Model Assumptions

There are n wireless nodes, some of which are legitimate and the rest adversarial. Prior to operation, each node is given a protocol p which is a set of instructions that describe the precise action that the node should take in response to external stimuli and its internal state. The legitimate nodes, by definition, are compelled to obey these instructions exactly. On the other hand, the adversarial nodes, after having obtained the protocol p, elect to follow a different set of instructions q_p, that by design undermine the operation of the network. The legitimate nodes are half-duplex and do not know which nodes are adversarial a priori. The adversarial nodes however, know the identities of their compatriots, and are able to fully and instantaneously coordinate all of their activity together via back channels of infinite bandwidth.

At time $t = 0$ the nodes, both legitimate and adversarial, turn on, and subsequently execute p and q_p respectively over a fixed operating lifetime. The interaction between the nodes results in an effective throughput vector x between *all* source-destination pairs. Each element of the vector x corresponds to the effective end-to-end throughput of a single source destination pair. Since there are n nodes, x has dimension $n(n-1)$. We say that $x = f(p, q_p)$ where f is defined by the model yet to be described. The utility function $U(x)$ defines the benefit or *utility* accrued to the network by operating at x.

We introduce the following zero-sum game between protocols and adversarial strategies. First the protocol p is announced to all nodes. Then the adversarial nodes choose a strategy q_p. The payoff of this game, to the legitimate nodes $J(p, q_p)$ is defined as $J(p, q_p) := U(x)$. Given this zero-sum game, the protocol that achieves the best possible payoff is the solution to the max-min optimization:

$$\max_{\text{protocols } p} \min_{\text{attacks } q_p} J(p, q_p), \tag{1}$$

This formulation, however, as a solution to the problem of designing secure protocols, is unsatisfactory; the utility in the payoff is measured over a throughput vector that includes adversarial source-destination pairs, whereas we would prefer to maximize the utility to those source-destination pairs that are exclusively legitimate. Since the legitimate nodes do not know which nodes are

adversarial a priori, the protocol will have to identify the adversarial nodes during the execution itself. The ability to discern the adversarial nodes from the legitimate nodes, depends on the actions of the adversarial nodes and the model assumptions. For now, we will defer from improving upon (1), until these assumptions have been stated. First, we will introduce a few concepts and definitions.

A *modulation scheme* specifies the carrier frequency, a symbol in the modulation alphabet, and transmit power level. Each modulation scheme $m \in \mathcal{M}_i$ used by node i corresponds to a message and an effective rate $r(m_i)$ at which the message is transmitted. The adversarial nodes are free to avail themselves of these legitimate modulation schemes, but could also choose instead to "jam"; emit a stream of random noise into the channel and disrupt the communication of neighboring nodes. Hence the set of modulation schemes from which an adversarial node can choose, in addition to the legitimate schemes, also includes schemes that specify a jamming frequency and transmit power level. In other words, a modulation scheme describes the physical action of a node, be it adversarial or legitimate, in the wireless channel.

A *concurrent transmission vector*, $c := \{m_1, \ldots, m_{n(n-1)}\}$, is defined as a vector of edges, where each edge specifies a source modulation scheme in the set of $n(n-1)$ one-hop, source-destination pairs.

A concurrent transmission vector is *feasible*, by definition, if the message transmitted by every edge in the vector is successfully received at the corresponding destination. When all the nodes are legitimate, feasibility is determined exclusively by the properties of the physical channel, and is equivalent to the notion of an independent set of edges, wherein each edge can be simultaneously active without disrupting the others.

The adversarial nodes can further augment this feasible set by sending messages through the back channel of infinite bandwidth instead of through the physical channel. Therefore the set of feasible concurrent transmission vectors includes both vectors that are feasible because of the physical channel, and those that are "artificially" feasible because of the assistance of adversarial nodes. We denote by $\mathcal{F}$, the set of feasible concurrent transmission vectors.

We say that $c_k \subset c_j$ if all of the edges of positive rate in c_k are also in c_j. We will assume that whenever a concurrent transmission vector c is feasible, the subsets of c are also feasible.

Some of the edges in a feasible concurrent transmission vector can be *disabled* if the adversarial nodes jam or fail to transmit the messages required of them. To make this concept more precise, we first introduce some additional notation. Let G and B denote the set of legitimate (good) and adversarial (bad) nodes respectively. Given a modulation scheme c, let c_{GG} and c_{BB} denote the vector of modulation schemes corresponding to the edges in c with legitimate and adversarial endpoints respectively. Similarly, let c_{GB} and c_{BG} denote the vector of modulation schemes corresponding to edges in c with a legitimate (adversarial) source and adversarial (legitimate) destination respectively. Therefore c can be equivalently represented as $c := (c_{GG}, c_{GB}, c_{BG}, c_{BB})$.

Formally, we say an edge in a feasible c can be *actively disabled* by adversarial nodes if there exists another concurrent transmission vector $c' := (c_{GG}, c_{GB}, c'_{BG}, c'_{BB})$ in which an edge in c' fails to receive the same set of messages at the same rates specified in c. For example, suppose $c_{GB} = c_{BG} = c_{BB} = \bar{0}$. Then an edge in c_{GG} can be actively disabled, by definition, if the adversarial nodes are able to block the corresponding message by jamming according to (c'_{BG}, c'_{BB}) instead of remaining silent as stipulated by (c_{BG}, c_{BB}). In both cases, the fate of the edge is determined by the physical properties of the channel.

In the more general case, $c_{GB}, c_{BG}, c_{BB} \neq \bar{0}$, where the adversarial nodes have messages of their own to transmit or receive, there are other possible ways in which edges can be disabled. For instance, an adversarial node could falsely deny receiving a message from a legitimate node, or conversely, claim to have sent a message to a legitimate node that was never received. The adversarial nodes can also disable edges in those concurrent transmission vectors that are artificially feasible, by failing to relay messages transmitted by legitimate nodes. Each of these outcomes is independent of a specific c', but our formal definition still applies for *some* c'.

We say that a feasible concurrent transmission vector c can be actively disabled if any edge in c can be actively disabled. Note that if c can be disabled, it does not follow that all of the subsets of c can also be disabled. We denote by $\mathcal{D}$, the set of concurrent transmission vectors that can be actively disabled. The concurrent transmission vectors in $\mathcal{D}$ can be actively disabled regardless of the protocol in which they occur.

An edge in c can also be *passively* disabled, if the adversarial nodes, through clever or subtle attacks, are able to prevent a protocol from ever deploying c. For instance, the adversarial nodes could prevent the nodes from arriving at a common topological view or schedule, thus precluding the use of some concurrent transmission vectors. These types of attacks are tailored to specific protocols, and are not included in our formal definition of an actively disabled concurrent transmission vector; whenever some c is passively disabled we will explicitly say so. We are now ready to state the model assumptions.

(M1) There are a finite number of modulation schemes from which a legitimate node can choose to physically transmit a message.

It follows from (M1) that the set of all concurrent transmission vectors is also finite. We denote by $\mathcal{C} :=$

$\{c_j, j = 1, \ldots, N\}$ the set of all concurrent transmission vectors indexed from $1, \ldots, N$.

(M2) The legitimate nodes are connected. More precisely, there exists an edge set $\mathcal{E}$ with the following properties: the subgraph of *all* legitimate nodes in $\mathcal{E}$ is connected, the edges are of positive rate $r(e_j) > 0$, and the edges cannot be actively disabled. Naturally, the legitimate nodes do not know a priori which edges belong to this specially designated set.

This assumption, that the legitimate nodes form a connected subgraph over which messages are conveyed with with complete reliability, does not accurately capture the dynamics of channel fading or probabilistic packet loss. We will address this concern when discussing the limitations of this work, and future areas of improvement.

(M3) The legitimate nodes can encrypt their messages and an encrypted packet can never be tampered with or forged. In addition, each node possesses an identity certificate provided by a trusted central authority, and a list of IDs that correspond to each node.

This assumption is reasonable given the growth of computational power making the relative expense of asymmetric encryption more affordable. Similarly, we will further assume that the individual nodes are able to process any tasks instantaneously.

(M4) Each node has a local clock, and the local clocks of the legitimate nodes tick at the same rate. Moreover, the legitimate nodes turn on simultaneously at $t = 0$ with respect to some global reference clock. Therefore the local clocks are perfectly synchronized.

III. A Game Between Protocols and Adversarial Strategies

We now attempt to modify the game proposed in (1) so that the payoff will be derived exclusively from utility accrued to legitimate source-destination pairs. The difficulty is that the legitimate nodes do not know which nodes are adversarial a priori, but must attempt to discern the adversarial nodes from their actions during the course of the protocol execution. We will attempt to exploit the premise of a connected legitimate subgraph in model assumption (M2) to remove from x any source-destination pairs that do not belong to this connected component. However, there are two challenges we must confront.

First, suppose some of the adversarial nodes decide to perfectly conform to the protocol, that is, suppose $q_p = p$. Such a strategy is justifiable if an adversarial node, located far away from the other nodes, is able to more effectively than by jamming, reduce throughput by drawing a disproportionate share of the allocated service time. If the adversarial nodes purporting to be legitimate, are part of a connected component consisting of legitimate nodes then the adversarial nodes, being indistinguishable from the legitimate nodes, can never be identified by any protocol.

The second obstacle is an extension of the first. Over the operating lifetime $[0, T)$, the execution of a protocol can yield multiple connected components consisting of legitimate and conforming adversarial nodes; at any instant $t \in [0, T)$, an adversarial node may stop conforming, and remove itself from the current component, leaving a smaller component in its wake. To determine the payoff, we will need to factor into consideration all the connected components in which the legitimate nodes take part, over the entire operating lifetime.

This information cannot be gleaned from the effective throughput x, but instead requires a nuanced view of the interaction between the protocol p and the adversarial response q_p at every time instant t. To provide a preview of what lies ahead, in Section III-A we will describe precisely, the throughput as a function of (p, q_p) over any time interval $[T_1, T_2) \subset [0, T)$. Next, in Section III-B we will introduce a definition of the connected component of legitimate nodes. Finally, in Section III-C we will define the payoff of the game as the time-weighted average of the utility accrued to each connected component of legitimate nodes, where the utility depends the effective throughput over the time intervals in which the connected component is active.

A. Deriving the Throughput Vector

Given a protocol p, and a concurrent transmission vector $c_j \in \mathcal{C}$, we set the indicator function $I_j^{(p)}(t) = 1$ for each time instant t in which p attempts to activate c_j. (Note that p may not succeed in this attempt if c_j is disabled, an issue we will address shortly.) We will assume by convention, that whenever $I_j^{(p)}(t) = 1$, all of the subsets of c_j are simultaneously in play as well; $I_k^{(p)}(t) = 1$ for all $c_k \subset c_j$.

We will denote by $q_p(t)$, the set of feasible concurrent transmission vectors both actively *and* passively disabled at time t, by the adversarial response q_p. Recall that a concurrent transmission vector is disabled (either actively or passively), by definition, if even one participating edge is disabled. Therefore if c_j is disabled it does *not* follow that all the subsets of c_j are also disabled.

The next definition is more subtle. We say that the concurrent transmission vector c_j is *enabled* at time t, by definition, if c_j is the *maximal* concurrent transmission vector to satisfy the following conditions: c_j is feasible, $I_j^{(p)}(t) = 1$, and $c_j \notin q_p(t)$. By maximal we mean there is no c_k, where $c_j \subset c_k$, that satisfies the preceding conditions. In other words, if c_j is enabled then by definition, the subsets of c_j are *not* enabled. We set the indicator function $\tilde{I}_j^{(p)}(t) = 1$ whenever c_j is enabled by the protocol p at time t. We denote by γ_j the fraction of the interval $[T_1, T_2)$ for which c_j is enabled. The throughput vector $x = f(p, q_p)$ over *any* time-interval $[T_1, T_2)$ satisfies the following constraints:

$\sum_{j=1}^{N} \gamma_j \leq 1$, $\sum_{i=j}^{N} \gamma_j r(c_j) = z$, $\sum_{u:(n,m)\in u} y_u \leq z_{(n,m)}$, $\sum_{u:u\in(s_i,d_i)} y_u = x_{(s_i,d_i)}$, $\gamma_j, y_u \geq 0$, $\gamma_j, y_u \geq 0$, and $\gamma_j := \frac{1}{T_2-T_1}\int_{T_1}^{T_2} \tilde{I}_j^{(p)}(t)\,dt$.

The first five constraints are standard in the network optimization literature [4], and basically state that the capacity region of throughput vectors is defined by the convex hull of feasible concurrent transmission vectors. The last equality follows from the definition of γ_j.

B. Defining a Connected Component

We will now introduce the definition of a connected component of legitimate nodes. At each time instant t, let $\mathcal{F}_t := \mathcal{F} \setminus q_p(t)$ denote the set of feasible concurrent transmission vectors that have not been disabled, either actively or passively by the adversarial strategy $q(t)$. Let $\mathcal{E}_t$ denote the set of edges in $\mathcal{F}_t$ that are either legitimate *or* part of a connected component of positive rate edges that includes at least one legitimate node. The connected component of legitimate nodes at time t is denoted by $C_t(p, q_p)$ and defined as the set of source-destination pairs that have edges in $\mathcal{E}_t$.

For any protocol p and adversarial response $q_p(t)$ there will be at most a finite number of distinct connected components $C_t(p, q_p)$ in the execution over the entire operating lifetime $[0, T)$ (since there are a finite number of nodes). We will index each connected component in the execution of (p, q_p) by $j = 1, \ldots, M$, where $C^{(j)}(p, q_p)$ denotes the jth connected component, and the value of M depends on the execution itself. We denote by α_j the fraction of the operating lifetime over which $C^{(j)}(p, q_p)$ is active. The throughput vector $x_{C^{(j)}(p,q_p)}$ denotes the effective throughput corresponding to each of the source-destination pairs in $C^{(j)}(p, q_p)$, evaluated over the intervals in which $C^{(j)}(p, q_p)$ is active, as per the constraints in Section III-A.

C. Defining the Payoff

We will now define the payoff of the game $J(p, q_p)$ as the time-averaged utility accrued to every connected component of legitimate nodes over the execution (p, q_p). More precisely:

$$J(p, q_p) := \sum_{j=1}^{M} \alpha_j U\left(x_{C^{(j)}(p,q_p)}\right). \tag{2}$$

One detail to be clarified; the utility function $U(x)$ must be defined over all vectors x of dimension less than or equal to $n(n-1)$ where n is the number of nodes. We formulate the problem of secure protocol design as the protocol solution p^* to the max-min optimization problem (1) where $J(p, q_p)$ is defined in (2).

We conclude this section with a comment about whether the payoff $J(p, q_p)$, as we have defined it, accurately reflects the performance of the protocol. A notable feature of $J(p, q_p)$ is that the connected components over which the utility is evaluated include adversarial nodes. The payoff in (2) does not explicitly penalize the legitimate nodes for losing service time to the adversarial nodes in the connected component, nor does it provide an explicit incentive for the adversarial nodes to draw away service time from the legitimate nodes. We will leave this issue to be addressed in future work.

Algorithm 1 The Protocol

```
NEIGHBOR DISCOVERY PHASE
NETWORK DISCOVERY PHASE
for i = 1 ... n_iter do
    SCHEDULING PHASE
    DATA TRANSFER PHASE
    VERIFICATION PHASE
end for
```

IV. The Protocol

We now describe the protocol at the heart of this paper. The strategy of the protocol is to progressively whittle down the concurrent transmission vectors being considered. The protocol, as shown in Algorithm 1, is composed of five phases: a neighbor discovery phase, a network discovery phase, a scheduling phase, a data transfer phase, and a verification phase. The first two phases enable the network to form a rudimentary network in which the legitimate nodes share a common view of the network topology. The remaining three phases are part of an iterative process that, through the "whittling down" strategy described earlier, enables the network to converge towards a min-max utility-optimal throughput vector. To simplify the presentation, we will assume the viewpoint of a legitimate node n_A, in the aftermath of its primordial birth. We will describe, both, the progression of this node through the protocol phases, and its corresponding transformation from a solitary node to a participant in a functioning network that yields a min-max payoff-optimal throughput.

A. The Neighbor Discovery Phase

At time $t = 0$ with respect to some global reference clock, node n_A and the other legitimate nodes simultaneously turn on and enter the neighbor discovery phase. We will assume that the legitimate nodes are half-duplex, so all transmissions to follow must be scheduled to avoid primary and secondary conflicts. A primary conflict occurs when a half-duplex node is forced to simultaneously transmit one message and receive another, whereas a secondary conflict occurs when a node is forced to simultaneously receive two (or more) messages from two different sources. Since the network topology is unknown a priori, node n_A needs to use an orthogonal MAC code, or some equivalent method, to prevent such conflicts from occurring. This task is not burdensome since the legitimate nodes are effectively synchronized.

Upon entry, node n_A repeatedly broadcasts "probe" packets to the surrounding area. Each probe packet contains a certificate from a trusted authority confirming the identity of node n_A. While broadcasting its own probe packets, node n_A also listens for probe packets transmitted by other nodes. Upon receiving an external probe packet, node n_A through a handshake mechanism, creates a mutually authenticated link certificate with the corresponding neighbor. By the end of the neighbor discovery phase, node n_A is at a minimum, guaranteed to have mutually authenticated link certificates with each of its neighbors in the underlying connected graph of legitimate nodes (see (M2)). Node n_A then enters the network discovery phase.

B. The Network Discovery Phase

During the network discovery phase, node n_A shares its list of neighbors obtained in the preceding phase, with the rest of the network and receives corresponding lists of neighbors from the legitimate nodes as well as a subset of the adversarial nodes in return.

Given the same set of lists of neighbors, the legitimate nodes can infer the same topological view, derive the same schedule, and execute the derived schedule as a single coordinated entity. The challenge is in the first part; ensuring the legitimate nodes arrive at a consensus on the collection of lists of neighbors even if the adversarial nodes selectively distribute the lists to undermine the very consensus that must be achieved. This consensus problem was first proposed in [5] and we will briefly summarize the main results; they play a critical role in this phase and in the verification phase to follow. Suppose node n_A receives a list of neighbors L, that has been authenticated by the following sequence of nodes in the order given: $n_B \rightarrow n_C \rightarrow n_D$. Then node n_A knows that nodes n_B, n_C, and n_D have all obtained L. But node n_A also knows more; that node n_D knows that node n_C has obtained L. Node n_C, however, may not know that node n_D or node n_A have obtained L. To achieve consensus, it is not enough for nodes n_A, n_B, n_C and n_D to know L; they need to know that everyone knows they know L, and so forth. The seminal contribution of [5] is that this type of knowledge, known as common knowledge, can be achieved in a finite number of transmissions provided that certain combinations of conditions are satisfied, one being the existence of a connected component containing the legitimate nodes (M2) and perfect encryption (M3).

Therefore, contingent on model assumptions (M2) and (M3), the legitimate nodes in the network share a common topological view at the end of this process. Upon completion of the network discovery phase, this rudimentary network then begins the first of what will be many iterations of the scheduling, data transfer and verification phases.

C. The Scheduling Phase

We use the adjective "rudimentary" in the sense that the legitimate nodes do not know which concurrent transmission vectors in $\mathcal{C}$ are feasible or enabled. Instead the network starts with an estimate of the feasible enabled set which we will denote by $\mathcal{F}^{(1)}$, where the initial estimate includes all possible concurrent transmission vectors. That is, $\mathcal{F}^{(1)} := \mathcal{C}$. Let $\mathcal{E}^{(1)}$ denote the set of edges in $\mathcal{F}^{(1)}$ that have positive rate, and let $C^{(1)}$ denote the connected component of $\mathcal{E}^{(1)}$ that includes node n_A. Note that every legitimate node can, on the basis of (M2), correctly identify the legitimate connected component $\mathcal{E}^{(1)}$ as the component in which it is included. Since we are starting with $\mathcal{C}$ as an estimate, $C^{(1)}$ will include all nodes. In later iterations, after successive refinements, $C^{(k)}$ will hone in on the connected component that includes all legitimate nodes.

The legitimate node n_A determines a utility-optimal schedule over $C^{(1)}$ and proceeds to execute the schedule in the data-transfer phase.

D. The Data Transfer Phase

This schedule, based on an extremely coarse estimate of the feasible set of concurrent transmission vectors $\mathcal{F}^{(1)}$, possibly includes concurrent transmission vectors that are not feasible. In addition, the adversarial nodes, by withholding cooperation, may effectively disable certain concurrent transmission vectors. As a result the first data transfer phase is likely to be a total failure, with service time assigned to non-feasible or actively disabled concurrent transmission vectors, and many or even all of the scheduled packets failing to arrive at their respective destinations. However, the effort is not a complete waste, for each legitimate node makes a careful record of all the packets that failed to arrive as scheduled. The network then enters the verification phase.

E. The Verification Phase

The legitimate nodes proceed to share their lists of missing packets with the rest of the network, by way of a consensus algorithm similar to that used in the network discovery phase. The same arguments also guarantee that the legitimate nodes arrive at a common master list of missing packets, which includes the individual lists generated by each legitimate node. The network then infers from this list, the corresponding set of concurrent transmission vectors $\mathcal{D}^{(1)}$ that failed to deliver the missing packets. These concurrent transmission vectors are permanently removed from the updated estimate of feasible concurrent transmission vectors $\mathcal{F}^{(2)}$ in the upcoming iteration. That is, $\mathcal{F}^{(2)} := \mathcal{F}^{(1)} \setminus \mathcal{D}^{(1)}$. The crucial insight, contingent on (M2), is that $\mathcal{D}^{(1)}$ will never include a feasible, enabled concurrent transmission vector. Upon completion of the verification phase, the network returns to the scheduling phase for the next iteration.

F. Steady State

The legitimate nodes, now one iteration "wiser", determine the schedule that supports a utility-optimal throughput based on the updated estimate $\mathcal{F}^{(2)}$ and proceed again to the data transfer phase. As before, each legitimate node keeps a record of all the scheduled packets it failed to receive. The lists are shared with the rest of the network during the verification phase, and the culpable concurrent transmission vectors $\mathcal{D}^{(2)}$ are pruned from $\mathcal{F}^{(3)}$, where $\mathcal{F}^{(3)} := \mathcal{F}^{(2)} \setminus \mathcal{D}^{(2)}$.

The process repeats for n_{iter} iterations where we will choose n_{iter} in the proof to be much larger than N (recall that N denotes the total number of concurrent transmission vectors in $\mathcal{C}$ and is a function of the number of nodes n and modulation schemes). After each iteration, the estimate of feasible concurrent transmission vectors is updated according to the rule $\mathcal{F}^{(k+1)} := \mathcal{F}^{(k)} \setminus \mathcal{D}^{(k)}$.

The protocol confines the options of the adversarial nodes after each iteration, to actively disabling concurrent transmission vectors that have not already been disabled. We will show in the proof that the effective throughput vector over the entire operating lifetime, is component-wise within ϵ of the min-max payoff-optimal throughput vector.

V. Main Results

The results here are based on the assumption that any concurrent transmission vector disabled by an adversarial strategy q_p, is permanently disabled; $q_p(s) \subset q_p(t)$ for all $s \leq t$. As an aside, we acknowledge that more general attack models can be considered in which concurrent transmission vectors disabled at time s can potentially be reactivated at time $t \geq s$. We will leave such models for future work.

Let (p_1^*, q_1^*) denote the solution to the min-max optimization problem in (1), and let x_1 denote the corresponding effective throughput over the entire operating lifetime $[0, T)$. That is, $x_1 := f(p_1^*, q_1^*)$. We have the following result:

Theorem 1. *Given an arbitrary $\epsilon > 0$, the protocol in Section IV enables the network to operate at an effective throughput of $(1 - \epsilon)x_1$.*

The max-min payoff in (1) would appear to be optimal given that according to the game, the protocol is always chosen before the adversarial strategy. However, we can do better. Let us revise the game so that the adversarial strategy q is chosen before the protocol p_q. Without knowing the protocol a priori, the adversarial nodes are strategically limited to actively disabling concurrent transmission vectors. That is $q(t) \subset \mathcal{D}$ for all $t \in [0, T)$. We denote this set of strategies by $\tilde{\mathcal{D}}$. We have the following optimization problem:

$$\min_{\text{attacks } q \in \tilde{\mathcal{D}}} \max_{\text{protocols } p} J(p_q, q). \tag{3}$$

Let (p_2^*, q_2^*) denote the solution to the optimization problem in (3), and let x_2 denote the corresponding effective throughput. That is, $x_2 := f(p_2^*, q_2^*)$. We have the following result:

Theorem 2. *Given an arbitrary $\epsilon > 0$, the protocol of Section IV enables the network to operate at an effective throughput of $(1 - \epsilon)x_2$.*

Note that Theorem 2 implies Theorem 1; in general the min-max value of a game is greater than or equal its max-min counterpart. Since the protocol is actually revealed before an adversarial strategy is selected, Theorem 2 implies there is a saddle-point between protocols and adversarial strategies in the game; the adversarial nodes gain no advantage from knowing the protocol a priori.

Also of significance is the fact that the adversarial strategies $\{q \in \mathcal{D}\}$, are limited to actively disabling concurrent transmission vectors; $q(t) \subset \mathcal{D}$. The concurrent transmission vectors in $\mathcal{D}$ are purely determined by the compliance of adversarial nodes and the properties of the physical channel and are independent of the protocol in which they are deployed. As a result, we can say from Theorem 2 that the adversarial nodes are effectively limited to jamming which cannot be prevented by any protocol. However, $\tilde{\mathcal{D}}$ also includes strategies reminiscent of the "partial-deafness" attack; adversarial nodes appear legitimate even if they disable some of the concurrent transmission vectors in $\mathcal{D}$ but not others. These types of attacks cannot be detected by the protocol we have described. It remains to be seen if there are protocols that can do better.

VI. Proof of Theorem 2

For ease of presentation, we divide the proof into three parts. First, we show that the legitimate nodes, from their primordial birth, form a rudimentary subnetwork in which they share a common topological view. Next we show that this rudimentary network will eventually operate at a max-min utility optimal throughput, before deducting the throughput loss from the protocol overhead. Finally, we show that the throughput loss can be made arbitrarily small.

Lemma 1. *By the end of the network discovery phase, the legitimate nodes share a common topological view.*

Proof. An adversarial node cannot assume the identity or simulate the transmissions of a legitimate node (M3). Moreover, the legitimate nodes share an underlying connected component that cannot be actively disabled (M2). It follows that the legitimate nodes will establish mutually authenticated link certificates with every legitimate neighbor in the underlying connected component

by the end of the neighbor discovery phase. To prove consensus, we need to show that the legitimate nodes will, by the end of the network discovery phase, have access to the same collection of link certificates. Here we refer to a fundamental result in distributed systems [5] that states a connected graph of synchronized legitimate nodes can achieve consensus given perfect encryption. These conditions are satisfied in (M2), (M3), and (M4). □

Let ϵ_l denote the long-term throughput loss incurred from the use of disabled and non-feasible concurrent transmission vectors over the operating lifetime. Let ϵ_s denote the short-term overhead loss incurred during a single iteration for the time allocated to the non data-transfer phases. We have the following lemma:

Lemma 2. *The effective throughput over the operating lifetime, of the network under the protocol is* $(1-\epsilon_l)(1-\epsilon_s)x_2$.

Proof. Since the legitimate nodes share a common schedule and in the scheduling phase, share a common view of both the topology and the disabled concurrent transmission vectors, the adversarial nodes are effectively confined to *actively* disabling concurrent transmission vectors. Let us denote one such strategy $q \in \tilde{\mathcal{D}}$. Let n_{iter} and N denote the number of protocol iterations and concurrent transmission vectors respectively. Since disabled concurrent transmission vectors are iteratively pruned and $q(s) \subset q(t)$ for all $s \leq t$ (by assumption), it follows that for at least $n_{iter} - N$ iterations, i_k, where $k = 1, \ldots, n_{iter} - N$ and $i_k \in \{1, \ldots, n_{iter}\}$ we have $\mathcal{F}^{(i_k)} := \mathcal{F} \setminus q(t)$ during all time intervals in which these iterations occur. We choose n_{iter} so that $\epsilon_l = \frac{N}{n_{iter}}$. Let $x^*_{C^{(i_k)}(p,q)}$ denote the utility-optimal throughput over the connected component $C^{(i_k)}(p,q)$ during the i_kth iteration. By design, the network operates at $(1-\epsilon_s)x^*_{C^{(i_k)}(p,q)}$ for all $k = 1, \ldots, n_{iter} - N$. These iterations make up $(1-\epsilon_l)$ of the total. Therefore the effective throughput over the entire operating lifetime is $(1-\epsilon_l)(1-\epsilon_s)x$ where $x = f(p^*, q)$ and p^* is the solution to $\max_p J(p,q)$. The lemma follows by noting that $\max_p J(p,q) \geq \min_{q \in \tilde{\mathcal{D}}} \max_p J(p,q)$ and that $x_2 := f(p^*, q^*)$, where q^* is the solution to $\min_{q \in \tilde{\mathcal{D}}} \max_p J(p,q)$. □

Finally, we show that throughput loss can be made arbitrarily small.

Lemma 3. *The long-term throughput loss ϵ_l and the short-term throughput loss ϵ_s can be made arbitrarily small.*

Proof. Recall that the throughput loss ϵ_l denotes the loss incurred from using disabled or non-feasible concurrent transmission vectors. For ϵ_l to be arbitrarily small, the number of protocol iterations must be much larger than the number of concurrent transmission vectors. That is, $\epsilon_l := \frac{N}{n_{iter}}$. The short-term throughput loss over a protocol iteration, ϵ_s, requires more attention. We need to choose the duration of the data transfer phase D to be much larger than the combined duration of the neighbor discovery, network discovery, scheduling, and verification phases; the protocol overhead. These phases depend on the number of bits that are exchanged during the course of one protocol iteration which in turn depends on the the *network constants* and the *free parameters*. The first denotes intrinsic features of the network that cannot be altered by the protocol such as the number of nodes and the size of the encryption keys. The second refers to features of the protocol that can be altered to achieve a desired performance target. In this paper, this feature is the operating lifetime T, and the corresponding target is an arbitrary short-term throughput loss ϵ_s. The packet sizes in the overhead phases depend on the operating lifetime T through the time-stamps carried by the packets, which are of size $K_1 \log T$, where K_1 is a function of the network constants alone. The number of packets in the overhead phases K_2 is also determined by the network constants. It follows that the protocol overhead, measured in bits is given by $K_1 K_2 \log T$. Now choose D and T to satisfy the two constraints: $\epsilon_s \geq \frac{D}{D + K_2 K_1 \log T}$, and $T \geq n_{iter}(K_2 K_1 \log T + D)$ The first constraint, ensures that the data transfer phase is long enough to amortize the throughput loss from the overhead phases. The second constraint ensures that the operating lifetime is long enough to accommodate n_{iter} protocol iterations each of duration $K_2 K_1 \log T + D$. It is straightforward to show that a suitable choice of D and T exist. Therefore ϵ_l and ϵ_s can be made arbitrarily small. □

VII. Concluding Remarks

We proposed a new approach to the design of secure wireless protocols that offers comprehensive and provable security guarantees.

We reiterate two possible areas of improvement: a communication model that includes probabilistic packet loss and channel fading, and a payoff function for the game that captures more sophisticated adversarial strategies like the partial-deafness attack.

References

[1] NSA Information Assurance Solutions Group. Defense in depth. [Online]. Available: https://www.nsa.gov/ia/_files/support/defenseindepth.pdf

[2] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," *Wirel. Netw.*, vol. 11, no. 1-2, pp. 21–38, Jan. 2005.

[3] J. Ponniah, Y. Hu, and P. R. Kumar, "A system-theoretic clean slate approach to provably secure ad hoc wireless networking," *IEEE Transactions on Control of Network Systems, To appear*.

[4] F. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, pp. 33–37, 1997.

[5] L. Lamport, R. Shostak, and M. Pease, "The byzantine generals problem," *ACM Trans. Program. Lang. Syst.*, vol. 4, no. 3, pp. 382–401, Jul. 1982.