IEEE TRANSACTIONS ON MOBILE COMPUTING VOL.12 NO.2 YEAR 2013

Vampire attacks: Draining life from wireless ad-hoc sensor networks

Eugene Y. Vasserman* and Nicholas Hopper
Kansas State University University of Minnesota
eyv@ksu.edu hopper@cs.umn.edu

*Work performed while at the University of Minnesota

***Abstract*—Ad-hoc low-power wireless networks are an exciting research direction in sensing and pervasive computing. Prior security work in this area has focused primarily on denial of communication at the routing or medium access control levels. This paper explores resource depletion attacks at the routing protocol layer, which permanently disable networks by quickly draining nodes' battery power. These "Vampire" attacks are not specific to any specific protocol, but rather rely on the properties of many popular classes of routing protocols. We find that all examined protocols are susceptible to Vampire attacks, which are devastating, difficult to detect, and are easy to carry out using as few as one malicious insider sending only protocol-compliant messages. In the worst case, a single Vampire can increase network-wide energy usage by a factor of $O(N)$, where N in the number of network nodes. We discuss methods to mitigate these types of attacks, including a new proof-of-concept protocol that provably bounds the damage caused by Vampires during the packet forwarding phase.**

***Index Terms*—Denial of service, security, routing, ad-hoc networks, sensor networks, wireless networks.**

I. Introduction

Ad-hoc wireless sensor networks (WSNs) promise exciting new applications in the near future, such as ubiquitous on-demand computing power, continuous connectivity, and instantly-deployable communication for military and first responders. Such networks already monitor environmental conditions, factory performance, and troop deployment, to name a few applications. As WSNs become more and more crucial to the everyday functioning of people and organizations, availability faults become less tolerable — lack of availability can make the difference between business as usual and lost productivity, power outages, environmental disasters, and even lost lives; thus high availability of these networks is a critical property, and should hold even under malicious conditions. Due to their ad-hoc organization, wireless ad-hoc networks are particularly vulnerable to denial of service (DoS) attacks [75], and a great deal of research has been done to enhance survivability [2, 5, 13, 14, 50, 75].

While these schemes can prevent attacks on the short-term availability of a network, they do not address attacks that affect long-term availability — the most permanent denial of service attack is to entirely deplete nodes' batteries. This is an instance of a *resource depletion attack*, with battery power as the resource of interest. In this paper we consider how routing protocols, even those designed to be secure, lack protection from these attacks, which we call *Vampire attacks*, since they drain the life from networks nodes. These attacks are distinct from previously-studied DoS, reduction of quality (RoQ), and routing infrastructure attacks as they *do not disrupt immediate availability*, but rather work over time to entirely disable a network. While some of the individual attacks are simple, and power-draining and resource exhaustion attacks have been discussed before [53, 59, 68], prior work has been mostly confined to other levels of the protocol stack, e.g. medium access control (MAC) or application layers, and to our knowledge *there is little discussion, and no thorough analysis or mitigation, of routing-layer resource exhaustion attacks*.

Vampire attacks are *not protocol-specific*, in that they do not rely on design properties or implementation faults of particular routing protocols, but rather *exploit general properties of protocol classes* such as link-state, distance-vector, source routing, and geographic and beacon routing. Neither do these attacks rely on flooding the network with large amounts of data, but rather try to transmit as little data as possible to achieve the largest energy drain, preventing a rate limiting solution. Since Vampires use *protocol-compliant messages*, these attacks are very difficult to detect and prevent.

Contributions. This paper makes three primary contributions. First, we thoroughly evaluate the vulnerabilities of existing protocols to routing layer battery depletion attacks. We observe that security measures to prevent Vampire attacks are *orthogonal to those used to protect routing infrastructure*, and so existing secure routing protocols such as Ariadne [29], SAODV [78], and SEAD [28] do not protect against Vampire attacks. Existing work on secure routing attempts to ensure that adversaries cannot cause path discovery to return an invalid network path, but Vampires do not disrupt or alter discovered paths, instead using existing valid network paths and protocol-compliant messages. Protocols that maximize power efficiency are also inappropriate, since they rely on cooperative node behavior and cannot optimize out malicious action. Second, we show simulation results quantifying the performance of several representative protocols in the presence of a single Vampire (insider adversary). Third, we modify an existing sensor network routing protocol to provably bound the damage from Vampire attacks during packet forwarding.

A. Classification

The first challenge in addressing Vampire attacks is defining them — what actions in fact constitute an attack? DoS attacks

in wired networks are frequently characterized by *amplification* [52, 54]: an adversary can amplify the resources it spends on the attack, e.g. use one minute of its own CPU time to cause the victim to use ten minutes. However, consider the process of routing a packet in any multi-hop network: a source composes and transmits it to the next hop toward the destination, which transmits it further, until the destination is reached, consuming resources not only at the source node but also at every node the message moves through. *If we consider the cumulative energy of an entire network, amplification attacks are always possible*, given that an *adversary can compose and send messages* which are processed by each node along the message path. So, the act of sending a message is in itself an act of amplification, leading to resource exhaustion, as long as the aggregate cost of routing a message (at the intermediate nodes) is lower than the cost to the source to compose and transmit it. So, we must drop amplification as our definition of maliciousness and instead focus on the cumulative energy consumption *increase* that a malicious node can cause while sending the same number of messages as an honest node.

We define a Vampire attack as the composition and transmission of a message that causes more energy to be consumed by the network than if an honest node transmitted a message of identical size to the same destination, although using different packet headers. We measure the strength of the attack by the ratio of network energy used in the benign case to the energy used in the malicious case, i.e. *the ratio of network-wide power utilization with malicious nodes present to energy usage with only honest nodes* when the number and size of packets sent remains constant. Safety from Vampire attacks implies that this ratio is 1. Energy use by malicious nodes is not considered, since they can always unilaterally drain their own batteries.

B. Protocols and assumptions

In this paper we consider the effect of Vampire attacks on link-state, distance-vector, source routing, and geographic and beacon routing protocols, as well as a logical ID-based sensor network routing protocol proposed by Parno *et al.* [53]. While this is by no means an exhaustive list of routing protocols which are vulnerable to Vampire attacks, *we view the covered protocols as an important subset of the routing solution space*, and stress that our attacks are likely to apply to other protocols.

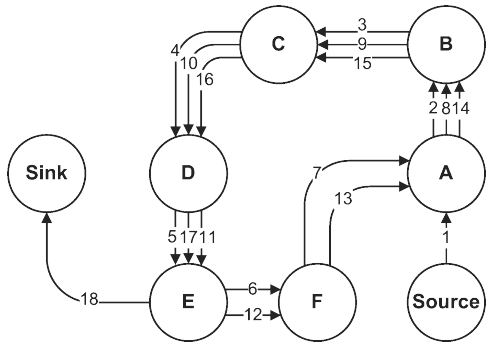
All routing protocols employ at least one topology discovery period, since ad-hoc deployment implies no prior position knowledge. Limiting ourselves to immutable but dynamically-organized topologies, as in most wireless sensor networks, we further differentiate on-demand routing protocols, where topology discovery is done at transmission time, and static protocols, where topology is discovered during an initial setup phase, with periodic re-discovery to handle rare topology changes. Our adversaries are malicious insiders and have the same resources and level of network access as honest nodes. Furthermore, adversary location within the network is assumed to be fixed and random, as if an adversary corrupts a number of honest nodes before the network was deployed, and cannot control their final positions. Note that this is far from the strongest adversary model; rather *this configuration represents the average expected damage from Vampire attacks*. Intelligent adversary placement or dynamic node compromise would make attacks far more damaging.

While for the rest of this paper we will assume that a node is permanently disabled once its battery power is exhausted, let us briefly consider nodes that recharge their batteries in the field, using either continuous charging or switching between active and recharge cycles. In the continuous charging case, power-draining attacks would be effective only if the adversary is able to consume power at least as fast as nodes can recharge. Assuming that packet processing drains at least as much energy from the victims as from the attacker, a continuously-recharging adversary can keep at least one node permanently disabled at the cost of its own functionality. However, recall that sending any packet automatically constitutes amplification, allowing few Vampires to attack many honest nodes. We will show later that a single Vampire may attack every network node simultaneously, meaning that continuous recharging does not help unless Vampires are more resource-constrained than honest nodes. Dual-cycle networks (with mandatory sleep and awake periods) are equally vulnerable to Vampires during active duty as long as the Vampire's cycle switching is in sync with other nodes. Vampire attacks may be weakened by using groups of nodes with staggered cycles: only active-duty nodes are vulnerable while the Vampire is active; nodes are safe while the Vampire sleeps. However, this defense is only effective when duty cycle groups outnumber Vampires, since it only takes one Vampire per group to carry out the attack.

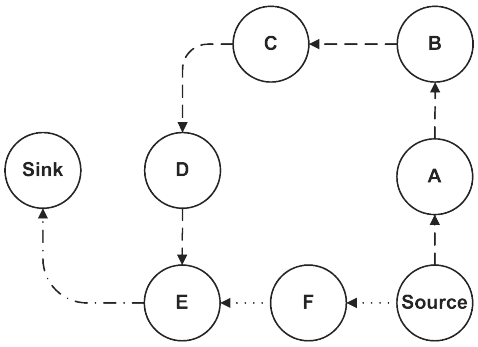
C. Overview

In the remainder of this paper, we present a series of increasingly damaging Vampire attacks, evaluate the vulnerability of several example protocols, and suggest how to improve resilience. In source routing protocols, we show how a malicious packet source can specify paths through the network which are far longer than optimal, wasting energy at intermediate nodes who forward the packet based on the included source route. In routing schemes where forwarding decisions are made independently by each node (as opposed to specified by the source), we suggest how directional antenna and wormhole attacks [30] can be used to deliver packets to multiple remote network positions, forcing packet processing at nodes that would not normally receive that packet at all, and thus increasing network-wide energy expenditure. Lastly, we show how an adversary can target not only packet forwarding but also route and topology discovery phases — if discovery messages are flooded, an adversary can, for the cost of a single packet, consume energy at every node in the network.

In our first attack, an *adversary composes packets with purposely introduced routing loops*. We call it the *carousel attack*, since it sends packets in circles as shown in Figure 1(a). It targets source routing protocols by exploiting the limited verification of message headers at forwarding nodes, allowing a single packet to repeatedly traverse the same set of nodes. Brief mentions of this attack can be found in other literature [10, 53], but no intuition for defense nor any evaluation is provided. In our second attack, also targeting



(a) An honest route would exit the loop immediately from node E to Sink, but a malicious packet makes its way around the loop twice more before exiting.



(b) Honest route is dotted while malicious route is dashed. The last link to the sink is shared.

Fig. 1. Malicious route construction attacks on source routing: carousel attack (a) and stretch attack (b).

source routing, *an adversary constructs artificially long routes*, potentially traversing every node in the network. We call this the *stretch attack*, since it increases packet path lengths, causing packets to be processed by a number of nodes that is independent of hop count along the shortest path between the adversary and packet destination. An example is illustrated in Figure 1(b). Results show that in a randomly-generated topology, a single attacker can use a carousel attack to increase energy consumption by as much as a factor of 4, while stretch attacks increase energy usage by up to an order of magnitude, depending on the position of the malicious node. The impact of these attacks can be further increased by combining them, increasing the number of adversarial nodes in the network, or simply sending more packets. Although in networks that do not employ authentication or only use end-to-end authentication, adversaries are free to replace routes in any overheard packets, *we assume that only messages originated by adversaries may have maliciously-composed routes*.

We explore numerous mitigation methods to bound the damage from Vampire attacks, and find that while the carousel attack is simple to prevent with negligible overhead, the stretch attack is far more challenging. The first protection mechanism we consider is *loose source routing*, where any forwarding node can reroute the packet if it knows a shorter path to the destination. Unfortunately, this proves to be less efficient than simply keeping global network state at each node, defeating the purpose of source routing. In our second attempt, we modify the protocol from [53] to guarantee that a packet makes progress through the network. We call this the *no-backtracking property*, since it holds if and only if a packet is moving strictly closer to its destination with every hop, and it mitigates all mentioned Vampire attacks with the exception of malicious flooded discovery, which is significantly harder to detect or prevent. We propose a limited topology discovery period ("the night," since this is when vampires are most dangerous), followed by a long packet forwarding period during which adversarial success is provably bounded. We also sketch how to further modify the protocol to detect Vampires during topology discovery and evict them after the network converges (at "dawn").

II. Related work

We do not imply that power draining itself is novel, but rather that these attacks have not been rigorously defined, evaluated, or mitigated at the routing layer. A very early mention of power exhaustion can be found in [68], as "sleep deprivation torture." As per the name, the proposed attack prevents nodes from entering a low-power sleep cycle, and thus deplete their batteries faster. Newer research on "denial-of-sleep" only considers attacks at the medium access control (MAC) layer [59]. Additional work mentions resource exhaustion at the MAC and transport layers [60, 75], but only offers rate limiting and elimination of insider adversaries as potential solutions. Malicious cycles (routing loops) have been briefly mentioned [10, 53], but no effective defenses are discussed other than increasing efficiency of the underlying MAC and routing protocols or switching away from source routing.

Even in non-power-constrained systems, depletion of resources such as memory, CPU time, and bandwidth may easily cause problems. A popular example is the SYN flood attack, wherein adversaries make multiple connection requests to a server, which will allocate resources for each connection request, eventually running out of resources, while the adversary, who allocates minimal resources, remains operational (since he does not intend to ever complete the connection handshake). Such attacks can be defeated or attenuated by putting greater burden on the connecting entity (e.g. SYN cookies [7], which offload the initial connection state onto the client, or cryptographic puzzles [4, 48, 73]). These solutions place minimal load on legitimate clients who only initiate a small number of connections, but deter malicious entities who will attempt a large number. Note that this is actually a form of rate limiting, and not always desirable as it punishes nodes who produce bursty traffic but may not send much total data over the lifetime of the network. Since Vampire attacks rely on amplification, such solutions may not be sufficiently effective to justify the excess load on legitimate nodes.

There is also significant past literature on attacks and defenses against quality of service (QoS) degradation, or reduction of quality (RoQ) attacks, that produce long-term degradation in network performance [23, 26, 41, 42, 44, 71, 76]. The focus of this work is on the transport layer rather

than routing protocols, so these defenses are not applicable. Moreover, since Vampires do not drop packets, the quality of the malicious path itself may remain high (although with increased latency).

Other work on denial of service in ad-hoc wireless networks has primarily dealt with adversaries who prevent route setup, disrupt communication, or preferentially establish routes through themselves to drop, manipulate, or monitor packets [14, 28, 29, 36, 78]. The effect of denial or degradation of service on battery life and other finite node resources has not generally been a *security consideration*, making our work tangential to the research mentioned above. Protocols that define security in terms of path discovery success, ensuring that only valid network paths are found, cannot protect against Vampire attacks, since Vampires do not use or return illegal routes or prevent communication in the short term.

Current work in minimal-energy routing, which aims to increase the lifetime of power-constrained networks by using less energy to transmit and receive packets (e.g. by minimizing wireless transmission distance) [11, 15, 19, 63], is likewise orthogonal: these protocols focus on cooperative nodes and not malicious scenarios. Additional on power-conserving medium access control (MAC), upper-layer protocols, and cross-layer cooperation [24, 34, 43, 45, 66, 67, 69, 77]. However, Vampires will increase energy usage even in minimal-energy routing scenarios and when power-conserving MAC protocols are used; these attacks cannot be prevented at the MAC layer or through cross-layer feedback. Attackers will produce packets which traverse more hops than necessary, so even if nodes spend the minimum required energy to transmit packets, each packet is still more expensive to transmit in the presence of Vampires. Our work can be thought of attack-resistant minimal-energy routing, where the adversary's goal includes decreasing energy savings.

Deng *et al.* discuss path-based DoS attacks and defenses in [13], including using one-way hash chains to limit the number of packets sent by a given node, limiting the rate at which nodes can transmit packets. While this strategy may protect against traditional DoS, where the malefactor overwhelms honest nodes with large amounts of data, it does not protect against "intelligent" adversaries who use a small number of packets or do not originate packets at all. As an example of the latter, Aad *et al.* show how *protocol-compliant* malicious intermediaries using intelligent packet-dropping strategies can significantly degrade performance of TCP streams traversing those nodes [2]. Our adversaries are also protocol-compliant in the sense that they use well-formed routing protocol messages. However, they either produce messages when honest nodes would not, or send packets with protocol headers different from what an honest node would produce in the same situation.

Another attack that can be thought of as path-based is the wormhole attack, first introduced in [30]. It allows two non-neighboring malicious nodes with either a physical or virtual private connection to emulate a neighbor relationship, *even in secure routing systems* [3]. These links are not made visible to other network members, but can be used by the colluding nodes to privately exchange messages. Similar tricks can be played using directional antennas. These attacks deny service by disrupting route discovery, returning routes that traverse the wormhole and may have artificially low associated cost metrics (such as number of hops or discovery time, as in rushing attacks [31]). While the authors propose a defense against wormhole and directional antenna attacks (called "Packet Leashes" [30]), their solution comes at a high cost and is not always applicable. First, one flavor of Packet Leashes relies on tightly synchronized clocks, which are not used in most off-the-shelf devices. Second, the authors assume that packet travel time dominates processing time, which may not be borne out in modern wireless networks, particularly low-power wireless sensor networks.

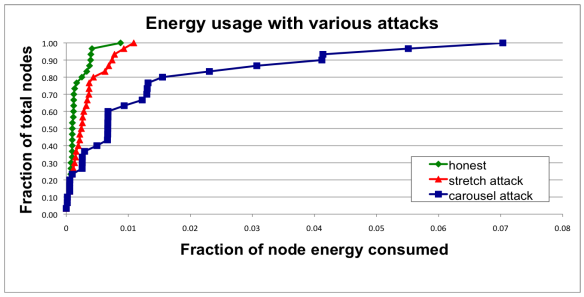


Fig. 2. Node energy distribution under various attack scenarios. The network is composed of 30 nodes and a single randomly-positioned Vampire. Results shown are based on a single packet sent by the attacker.

III. Attacks on Stateless Protocols

Here we present simple *but previously neglected* attacks on source routing protocols, such as DSR [35]. In these systems, the source node specifies the entire route to a destination within the packet header, so intermediaries do not make independent forwarding decisions, relying rather on a route specified by the source. To forward a message, the intermediate node finds itself in the route (specified in the packet header) and transmits the message to the next hop. The burden is on the source to ensure that the route is valid at the time of sending, and that every node in the route is a physical neighbor of the previous route hop. This approach has the advantage of requiring very little forwarding logic at intermediate nodes, and allows for entire routes to be sender-authenticated using digital signatures, as in Ariadne [29].

We evaluated both the carousel and stretch attacks (Figure 1(a)) in a randomly-generated 30-node topology and a single randomly-selected malicious DSR agent, using the ns-2 network simulator [1]. Energy usage is measured for the *minimum number* of packets required to deliver a single message, so sending more messages increases the strength of the attack linearly until bandwidth saturation.[1] We independently computed resource utilization of honest and malicious nodes and found that malicious nodes did not use a disproportionate amount of energy in carrying out the attack. In other words, malicious nodes are not driving down the cumulative energy of the network purely by their own use of energy. Nevertheless,

[1]Energy is debited from nodes only for packet transmission, and not for reception or processing, so the number of neighbors of the malicious node is not a confounding variable.

malicious node energy consumption data is omitted for clarity. The attacks are carried out by a randomly-selected adversary using the least intelligent attack strategy to obtain *average expected damage estimates*. More intelligent adversaries using more information about the network would be able to increase the strength of their attack by selecting destinations designed to maximize energy usage.

Per-node energy usage under both attacks is shown in Figure 2. As expected, the carousel attack causes excessive energy usage for a few nodes, since only nodes along a shorter path are affected. In contrast, the stretch attack shows more uniform energy consumption for all nodes in the network, since it lengthens the route, causing more nodes to process the packet. While both attacks significantly *network-wide* energy usage, individual nodes are also noticeably affected, with some losing almost 10% of their total energy reserve *per message*. Figure 3(a) diagrams the energy usage when node 0 sends a single packet to node 19 in an example network topology with only honest nodes. Black arrows denote the path of the packet.

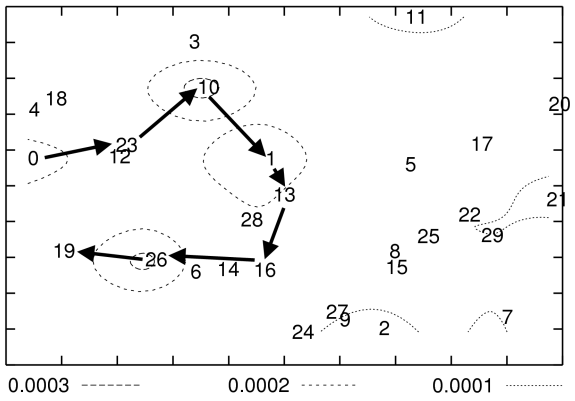
Carousel attack. In this attack, an *adversary sends a packet with a route composed as a series of loops*, such that the same node appears in the route many times. This strategy can be used to increase the route length beyond the number of nodes in the network, only limited by the number of allowed entries in the source route.[2] An example of this type of route is in Figure 1(a). In Figure 3(b), malicious node 0 carries out a carousel attack, sending a single message to node 19 (which does not have to be malicious). Note the drastic increase in energy usage along the original path.[3] Assuming the adversary limits the transmission rate to avoid saturating the network, the theoretical limit of this attack is an energy usage increase factor of $O(\lambda)$, where λ is the maximum route length.

Overall energy consumption increases by up to a factor of 3.96 per message. On average, a randomly-located carousel attacker in our example topology can increase network energy consumption by a factor of 1.48 ± 0.99. The reason for this large standard deviation is that the attack does not always increase energy usage — the length of the adversarial path is a multiple of the honest path, which is in turn, affected by the position of the adversary in relation to the destination, so the adversary's position is important to the success of this attack.

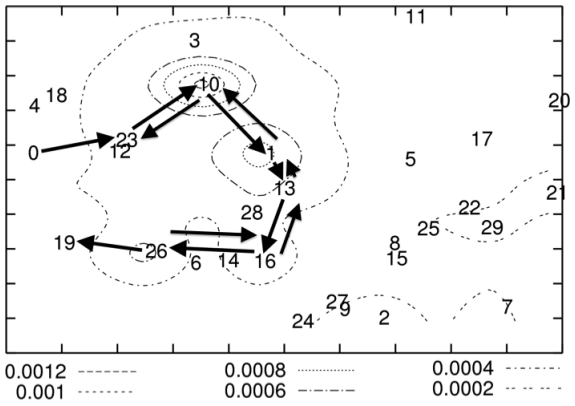
Stretch attack. Another attack in the same vein is the *stretch attack, where a malicious node constructs artificially long source routes*, causing packets to traverse a larger than optimal number of nodes. An honest source would select the route $Source \rightarrow F \rightarrow E \rightarrow Sink$, affecting four nodes including itself, but the malicious node selects a longer route, *affecting all nodes in the network*. These routes cause nodes that do not lie along the honest route to consume energy by forwarding packets they would not receive in honest scenarios. An example of this type of route is in Figure 1(b). The outcome becomes clearer when we examine Figure 3(c) and compare to the carousel attack. While the latter uses energy at the nodes who were already in the honest path, the former extends the consumed energy "equivalence lines" to a wider

[2]The ns-2 DSR implementation arbitrarily limits the route length to 16.

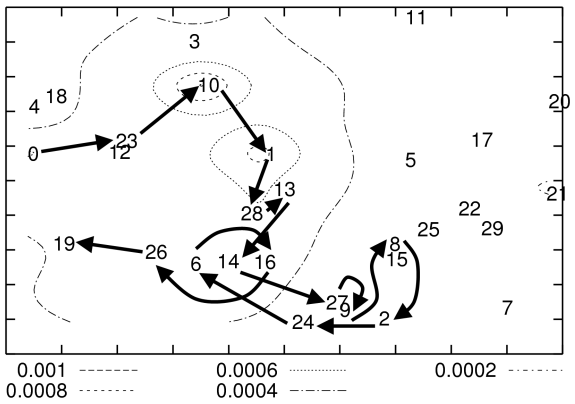
[3]Energy usage is greatest at node 10 likely due to its distance from its nearest neighbors.



(a) Honest scenario: node 0 sends a single message to node 19.

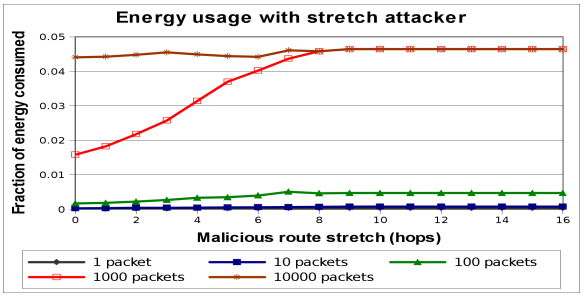


(b) Carousel attack (malicious node 0): the nodes traversed by the packet are the same as in (a), but the loop over all forwarding nodes roughly triples the route length (the packet traverses the loop more than once). Note the drastically increased energy consumption among the forwarding nodes.

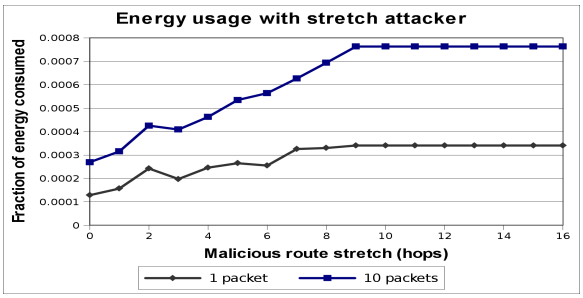


(c) Stretch attack (malicious node 0): the route diverts from the optimal path between source and destination, roughly doubling in length. Note that while the per-node energy consumption increase is not as drastic as in (b), the region of increased energy consumption is larger. Overall energy consumption is greater than in the carousel attack, but spread more evenly over more network nodes.

Fig. 3. Energy map of the network in terms of fraction of energy consumed per node. Black arrows show the packet path through the network. Each dotted line represents an "energy equivalence zone," similar to an area of equal elevation on a topological chart. Each line is marked with the energy loss by a node as a fraction of total original charge.



(a) A malicious node transmitting 1, 10, 100, 1,000, and 10,000 messages with artificially long paths.



(b) Zoomed Figure (a), displaying only the 1 and 10-message data.

Fig. 4. Effects of a single-node stretch attacker on a network of 30 nodes after removal of source route length limits. Maliciousness is measured in terms of the induced stretch of the optimal route, in number of hops.

section of the network. Energy usage is less localized around the original path, but more total energy is consumed.

The theoretical limit of the stretch attack is a packet that traverses every network node, causing an energy usage increase of factor $O(min(N, \lambda))$, where N is the number of nodes in the network and λ is the maximum path length allowed. This attack is potentially less damaging *per packet* than the carousel attack, as the number of hops per packet is bounded by the number of network nodes. However, adversaries can combine carousel and stretch attacks to keep the packet in the network longer: the resulting "stretched cycle" could be traversed repeatedly in a loop. Therefore, even if stretch attack protection is not used, route loops should still be detected and removed to prevent the combined attack. In our example topology, we see an increase in energy usage by as much as a factor of 10.5 *per message* over the honest scenario, with an average increase in energy consumption of 2.67 ± 2.49. As with the carousel attack, the reason for the large standard deviation is that the position of the adversarial node affects the strength of the attack. Not all routes can be significantly lengthened, depending on the location of the adversary. Unlike the carousel attack, where the *relative* positions of the source and sink are important, the stretch attack can achieve the *same effectiveness independent of the attacker's network position* relative to the destination, so the worst-case effect is far more likely to occur.

The true significance of the attack becomes evident in Figure 4(a), which shows network-wide energy consumption in the presence of a single randomly-selected Vampire in terms of the "maliciousness" of the adversary, or the induced stretch of the optimal route in number of hops. (Increasing maliciousness beyond 9 has no effect due to the diameter of our test topology.) Network links become saturated at 10,000 messages per second (even without the stretch attack), but the adversary can achieve the same effects by sending an order of magnitude fewer messages at a stretch attack maliciousness level of 8 or greater. This reduces cumulative network energy by 3%, or almost the entire lifetime of a single node. Therefore, the stretch attack increases the effectiveness of an adversary by an order of magnitude, reducing its energy expenditure to compose and transmit messages. With 100 messages, the result is less severe, but still pronounced: the network loses 1% of its total energy, or 9% of the lifetime of a single node. The effect becomes less visible when we look at 10 messages or fewer in Figure 4(b), but is still noticeable.

Since DSR uses hop count as a cost metric, constructing longer source routes could in fact *decrease* the amount of per-hop energy spent on sending packets if energy minimization protocols were used since shorter physical distances decrease required sending power, and thus battery drain. We construct long routes greedily, assuming global topology knowledge,[4] but attacks can be further optimized to consume more energy by considering relative node distances — given enough information, our adversary could construct not just longer but *maximum-energy* paths. Forwarding nodes using minimum-energy routing [15, 63, 66] could replace long distance transmissions with a number of shorter-distance hops, but the attack still works since the malicious path is longer, independent of in-network optimizations applied to it.

These attacks would be less effective in hierarchical networks, where nodes send messages to aggregators, who in turn sends it to other aggregators, which route it to a monitoring point. The described attacks are only valid within the network "neighborhood" of the adversarial node. If an adversary corrupts nodes intelligently or controls a small but non-trivial percentage of nodes, it can execute these attacks within individual network neighborhoods: a single adversary per neighborhood would disable the entire network. We discuss the notion of neighborhoods in more detail in Section VIII.

A. Mitigation methods

The carousel attack can be prevented entirely by having forwarding nodes check source routes for loops. While this adds extra forwarding logic and thus more overhead, we can expect the gain to be worthwhile in in malicious environments. The ns-2 DSR protocol does implement loop detection, but confusingly does not use it to check routes in forwarded packets.[5] When a loop is detected, the source route could be corrected and the packet sent on, but one of the attractive features of source routing is that the route can itself be signed by the source [29]. Therefore, it is better to simply drop the packet, especially considering that the sending node is likely malicious (honest nodes should not introduce loops). An alternate solution is to alter how intermediate nodes process the source route. To forward a message, a node must determine

[4]This can be obtained by probing routes to every host in the network.

[5]Routes are not discarded after they have been used, but are stored in a local cache in case the same destination will be used in the near future. Routes retrieved from the local cache are checked for loops before use.

the next hop by locating itself in the source route. If a node searches for itself from the destination backward instead from the source forward, any loop that includes the current node will be automatically truncated (the last instance of the local node will be found in the source route rather than the first). No extra processing is required for this defense, since a node must perform this check anyway — we only alter the way the check is done.

The stretch attack is more challenging to prevent. Its success rests on the forwarding node not checking for optimality of the route. If we call the no-optimization case "strict" source routing, since the route is followed exactly as specified in the header, we can define *loose source routing*, where intermediate nodes may replace part or all of the route in the packet header if they know of a better route to the destination. This makes it necessary for nodes to discover and cache optimal routes to at least some fraction of other nodes, partially defeating the as-needed discovery advantage. Moreover, caching must be done carefully lest a maliciously suboptimal route be introduced. We simulated the loose source routing defense using random-length suboptimal paths in randomly-generated network topologies of up to 1,000,000 nodes, with diameter 10–14. Results (Figure 5) demonstrate that the amount of node-local storage required to achieve reasonable levels of mitigation approaches global topology knowledge, defeating the purpose of using source routing. The dashed trendline represents the expected path length of rerouted packets if each node stores $\log N$ network paths, where N is the number of network nodes, while the solid trendline represents the majority of actual network paths in a loose source-routing setup. The number of nodes traversed by loose source-routed packets is suboptimal by at least a factor of 10, with some routes approaching a factor of 50. Only a few messages encountered a node with a better path to the destination than the originally-assigned long source route. Therefore we conclude that *loose source routing is worse than keeping global state* at every node.

Alternatively, we can bound the damage of carousel and stretch attackers by limiting the allowed source route length based on the expected maximum path length in the network, but we would need a way to determine the network diameter.[6] While there are suitable algorithms [40, 74], there has been very little work on whether they could yield accurate results *in the presence of adversaries*. If the number of nodes is known ahead of time, graph-theoretic techniques can be used to estimate the diameter.

Rate limiting may initially seem to be a good defense,but upon closer examination we see it is not ideal. It limits malicious sending rate, potentially increasing network lifetime, but that increase becomes the maximum expected lifetime, since adversaries will transmit at the maximum allowed rate. Moreover, sending rate is already limited by the size of nodes' receive queues in rate-unlimited networks (as seen in the 10,000 message scenario in Figure 4(a)). Rate limiting also potentially punishes honest nodes that may transmit large amounts of time-critical (bursty) data, but will send little data over the network lifetime.

[6]Packet time to live (TTL) also limits route length, but it is set by the malicious sender. Intermediate nodes may able to reset it to a "reasonable" value, but it is unclear how to discover that value.

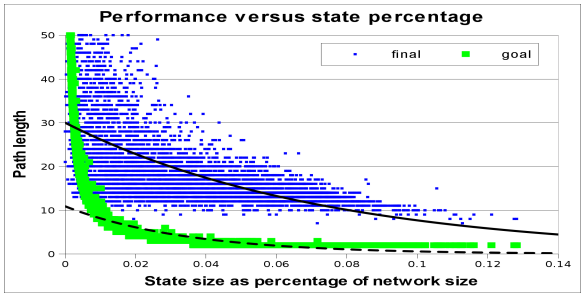


Fig. 5. Loose source routing performance compared to optimal, in a network with diameter slightly above 10. The dashed trendline represents expected path length when nodes store $\log N$ local state, and the solid trendline shows actual observed performance.

IV. Attacks on stateful protocols

We now move on to stateful routing protocols, where network nodes are aware of the network topology and its state, and make local forwarding decisions based on that stored state. Two important classes of stateful protocols are link-state and distance-vector. In link-state protocols, such as OLSR [12], nodes keep a record of the up-or-down state of links in the network, and flood routing updates every time a link goes down or a new link is enabled. Distance-vector protocols like DSDV [55] keep track of the next hop to every destination, indexed by a route cost metric, e.g. the number of hops. In this scheme, only routing updates that change the cost of a given route need to be propagated.

Routes in link-state and distance-vector networks are built dynamically from many independent forwarding decisions, so adversaries have limited power to affect packet forwarding, making these protocols immune to carousel and stretch attacks. In fact, any time adversaries cannot specify the full path, the potential for Vampire attack is reduced. However, malicious nodes can still mis-forward packets, forcing packet forwarding by nodes who would not normally be along packet paths. For instance, an adversary can forward packets either back toward the source if the adversary is an intermediary, or to a non-optimal next hop if the adversary is either an intermediary or the source. While this may seem benign in a dense obstacle-free topology,worst-case bounds are no better than in the case of the stretch attack on DSR. For instance, consider the special case of a ring topology: forwarding a packet in the reverse direction causes it to traverse every node in the network (or at least a significant number, assuming the malicious node is not the packet source but rather a forwarder), increasing our network-wide energy consumption by a factor of $O(N)$. While ring topologies are extremely unlikely to occur in practice, they do help us reason about worst-case outcomes. This scenario can also be generalized to routing around any network obstacle along a suboptimal path.

Directional antenna attack. Vampires have little control over packet progress when forwarding decisions are made independently by each node, but they can still waste energy by *restarting a packet* in various parts of the network. Using

a directional antenna adversaries can deposit a packet in arbitrary parts of the network, while also forwarding the packet locally. This consumes the energy of nodes that would not have had to process the original packet, with the expected additional honest energy expenditure of $O(d)$, where d is the network diameter, making $\frac{d}{2}$ the expected length of the path to an arbitrary destination from the furthest point in the network. This attack can be considered a half-wormhole attack [30], since a directional antenna constitutes a private communication channel, but the node on the other end is not necessarily malicious.[7] It can be performed more than once, depositing the packet at various distant points in the network, at the additional cost to the adversary for each use of the directional antenna. Packet Leashes cannot prevent this attack since they are not meant to protect against malicious message sources, only intermediaries [30].

Malicious discovery attack. Another attack on all previously-mentioned routing protocols (including stateful and stateless) is *spurious route discovery*. In most protocols, every node will forward route discovery packets (and sometimes route responses as well), meaning it is possible to initiate a flood by sending a single message. Systems that perform as-needed route discovery, such as AODV and DSR, are particularly vulnerable, since nodes may legitimately initiate discovery at any time, not just during a topology change. A malicious node has a number of ways to induce a *perceived* topology change: it may simply falsely claim that a link is down, or claim a new link to a non-existent node. Security measures, such as those proposed by Raffo *et al.* in [58] may be sufficient to alleviate this particular problem. Further, two cooperating malicious nodes may claim the link between them is down. However, nearby nodes might be able to monitor communication to detect link failure (using some kind of neighborhood update scheme). Still, short route failures can be safely ignored in networks of sufficient density. More serious attacks become possible when nodes claim that a long-distance route has changed. This attack is trivial in open networks with unauthenticated routes, since a single node can emulate multiple nodes in neighbor relationships [16], or falsely claim nodes as neighbors. Therefore, let us assume closed (Sybil-resistant) networks where link states are authenticated, similar to route authentication in Ariadne [29] or path-vector signatures in [70]. Now our adversary must present an *actually* changed route in order to execute the attack. To do this, two cooperating adversaries communicating through a wormhole could repeatedly announce and withdraw routes that use this wormhole, causing a theoretical energy usage increase of a factor of $O(N)$ per packet. Adding more malicious nodes to the mix increases the number of possible route announce/withdrawal pairs. Packet Leashes [30] cannot prevent this attack, with the reasoning being similar to the directional antenna attack — since the originators are themselves malicious, they would forward messages through the wormhole, and return only seemingly valid (and functional) routes in response to discovery. This problem is similar to route flapping in BGP [72], but while Internet paths are relatively stable [25, 61] paths change frequently in wireless ad-hoc networks, where nodes may move in and out of each other's range, or suffer intermittent environmental effects. Since there may be no stable routes in WSNs (hence the need for ad-hoc protocols), this solution would not be applicable.

[7]The attack is not very effective when using virtual wormholes (encrypted connections), since adversaries sending packets to each other would accomplish the same goal.

A. Coordinate and beacon-based protocols

Some recent routing research has moved in the direction of coordinate- and beacon-based routing, such as GPSR and BVR [21, 37], which use physical coordinates or beacon distances for routing, respectively. In GPSR, a packet may encounter a dead end, which is a localized space of minimal physical distance to the target, but without the target actually being reachable (e.g. the target is separated by a wall or obstruction). The packet must then be diverted (in GPSR, it follows the contour of the barrier that prevents it from reaching the target) until a path to the target is available. In BVR, packets are routed toward the beacon closest to the target node, and then move away from the beacon to reach the target. Each node makes independent forwarding decisions, and thus a Vampire is limited in the distance it can divert the packet. These protocols also fall victim to directional antenna attacks in the same way as link-state and distance-vector protocols above, leading to energy usage increase factor of $O(d)$ per message, where d is the network diameter. Moreover, GPSR does not take path length into account when routing around local obstructions, and so malicious mis-routing may cause up to a factor of $O(c)$ energy loss, where c is the circumference of the obstruction, in hops.

V. Clean-slate sensor network routing

In this section we show that a clean-slate secure sensor network routing protocol by Parno, Luk, Gaustad, and Perrig ("PLGP" from here on) [53] can be modified to provably resist Vampire attacks during the packet forwarding phase. The original version of the protocol, although designed for security, is vulnerable to Vampire attacks. PLGP consists of a topology discovery phase, followed by a packet forwarding phase, with the former optionally repeated on a fixed schedule to ensure that topology information stays current. (There is no on-demand discovery.) Discovery *deterministically* organizes nodes into a tree that will later be used as an addressing scheme. When discovery begins, each node has a limited view of the network — the node knows only itself. Nodes discover their neighbors using local broadcast, and form ever-expanding "neighborhoods," stopping when the entire network is a single group. Throughout this process, nodes build a tree of neighbor relationships and group membership that will later be used for addressing and routing.

At the end of discovery, each node should compute the same address tree as other nodes. All leaf nodes in the tree are physical nodes in the network, and their virtual addresses correspond to their position in the tree (see Figure 6). All nodes learn each others' virtual addresses and cryptographic keys. The final address tree is verifiable after network convergence, and all forwarding decisions can be independently verified.

Furthermore, assuming each legitimate network node has a unique certificate of membership (assigned before network deployment), nodes who attempt to join multiple groups, produce clones of themselves in multiple locations, or otherwise cheat during discovery can be identified and evicted.

Topology discovery. Discovery begins with a time-limited period during which every node must announce its presence by broadcasting a certificate of identity, including its public key (from now on referred to as node ID), signed by a trusted offline authority. Each node starts as its own group of size one, with a virtual address 0. Nodes who overhear presence broadcasts form groups with their neighbors. When two individual nodes (each with an initial address 0) form a group of size two, one of them takes the address 0, and the other becomes 1. Groups merge preferentially with the smallest neighboring group, which may be a single node. We may think of groups acting as individual nodes, with decisions made using secure multiparty computation. Like individual nodes, each group will initially choose a group address 0, and will choose 0 or 1 when merging with another group. Each group member prepends the group address to their own address, e.g. node 0 in group 0 becomes 0.0, node 0 in group 1 becomes 1.0, and so on. Each time two groups merge, the address of each node is lengthened by one bit. Implicitly, this forms a binary tree of all addresses in the network, with node addresses as leaved. Note that this tree is *not a virtual coordinate system*, as the only information coded by the tree are neighbor relationships among nodes.

Nodes will request to join with the smallest group in their vicinity, with ties broken by group IDs, which are computed cooperatively by the entire group as a deterministic function of individual member IDs. When larger groups merge, they both broadcast their group IDs (and the IDs of all group members) to each other, and proceed with a merge protocol identical to the two-node case. Groups that have grown large enough that some members are not within radio range of other groups will communicate through "gateway nodes," which are within range of both groups. Each node stores the identity of one or more nodes through which it heard an announcement that another group exists. That node may have itself heard the information second-hand, so every node within a group will end up with a next-hop path to every other group, as in distance-vector. Topology discovery proceeds in this manner until all network nodes are members of a single group. By the end of topology discovery, each node learns every other node's virtual address, public key, and certificate, since every group members knows the identities of all other group members and the network converges to a single group.

Packet forwarding. During the forwarding phase, all decisions are made independently by each node. When receiving a packet, a node determines the next hop by finding the most significant bit of its address that differs from the message originator's address (see Figure 6). Thus every forwarding event (except when a packet is moving within a group in order to reach a gateway node to proceed to the next group) shortens the *logical distance* to the destination, since node addresses should be strictly closer to the destination (see Function `forward_packet`).[8]

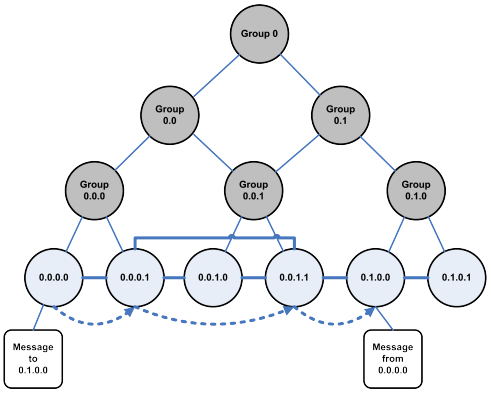


Fig. 6. The final address tree for a fully-converged 6-node network. Leaves represent physical nodes, connected with solid lines if within radio range. The dashed line is the progress of a message through the network. Note that non-leaf nodes *are not physical nodes* but rather logical group identifiers.

A. PLGP in the presence of Vampires

In PLGP, forwarding nodes do not know what path a packet took, allowing adversaries to divert packets to any part of the network, even if that area is logically further away from the destination than the malicious node. This makes PLGP vulnerable to Vampire attacks. Consider for instance the now-familiar directional antenna attack: a receiving honest node may be farther away from the packet destination than the malicious forwarding node, but the honest node has no way to tell that the packet it just received is moving away from the destination; the only information available to the honest node is its own address and the packet destination address, but not the address of the previous hop (who can lie). Thus, the Vampire can move a packet away from its destination *without being detected*. This packet will traverse at most $\log N$ logical hops, with $O(\sqrt{2^i})$ physical hops at the ith logical hop, giving us a theoretical maximum energy increase of $O(d)$, where d is the network diameter and N the number of network nodes. The situation is worse if the packet returns to the Vampire in the process of being forwarded — it can now be rerouted again, causing something similar to the carousel attack. Recall that the damage from the carousel attack is bounded by the maximum length of the source route, but in PLGP the adversary faces no such limitation, so the packet can cycle indefinitely. Nodes may sacrifice some local storage to retain a record of recent packets to prevent this attack from being carried out repeatedly with the same packet. Random direction vectors, as suggested in PLGP, would likewise alleviate the problem of indefinite cycles by avoiding the same malicious node during the subsequent forwarding round.

VI. Provable security against Vampire attacks

Here we modify the forwarding phase of PLGP to provably avoid the above-mentioned attacks. First we introduce the *no-backtracking property*, satisfied for a given packet if and only

[8]Since all node addresses are unique, every logical hop either takes a message strictly closer to, or further from, its destination.

Function `forward_packet(`p`)`

```
s ← extract_source_address(p);
c ← closest_next_node(s);
if is_neighbor(c) then forward(p, c);
else
    r ← next_hop_to_non_neighbor(c);
    forward(p, r);
```

Function `secure_forward_packet(`p`)`

```
s ← extract_source_address(p);
a ← extract_attestation(p);
if (not verify_source_sig(p)) or
(empty(a) and not is_neighbor(s)) or
(not saowf_verify(a)) then
    return ;                                  /* drop(p) */
foreach node in a do
    prevnode ← node;
    if (not are_neighbors(node, prevnode)) or
    (not making_progress(prevnode, node)) then
        return ;                              /* drop(p) */
c ← closest_next_node(s);
p' ← saowf_append(p);
if is_neighbor(c) then forward(p', c);
else forward(p', next_hop_to_non_neighbor(c));
```

if it consistently makes progress toward its destination in the logical network address space. More formally:

Definition 1. *No-backtracking is satisfied if every packet p traverses the same number of hops whether or not an adversary is present in the network. (Maliciously-induced route stretch is bounded to a factor of 1.)*

This does not imply that every packet in the network must travel the same number of hops regardless of source or destination, but rather that a packet sent to node D by a malicious node at location L will traverse the same number of hops as a packet sent to D by a node at location L that is honest. If we think of this in terms of protocol execution traces, no-backtracking implies that for each packet in the trace, the number of intermediate honest nodes traversed by the packet between source and destination is independent of the actions of malicious nodes. Equivalently, traces that include malicious nodes should show the same network-wide energy utilization by honest nodes as traces of a network with no malicious actors. The only notable exceptions are when adversaries drop or mangle packets en route, but since we are only concerned with packets initiated by adversaries, we can safely ignore this situation: "pre-mangled" packets achieve the same result — they will be dropped by an honest intermediary or destination.

No-backtracking implies Vampire resistance. It is not immediately obvious why no-backtracking prevents Vampire attacks in the forwarding phase. Recall the reason for the success of the stretch attack: intermediate nodes in a source route cannot check whether the source-defined route is optimal, or even that it makes progress toward the destination. When nodes make independent routing decisions such as in link-state, distance-vector, coordinate-based, or beacon-based protocols, packets cannot contain maliciously composed routes. This already means the adversary cannot perform carousel or stretch attacks — no node may unilaterally specify a suboptimal path through the network. However, a sufficiently clever adversary may still influence packet progress. We can prevent this interference by independently checking on packet progress: if nodes keep track of route "cost" or metric and, when forwarding a packet, communicate the local cost to the next hop, that next hop can verify that the remaining route cost is lower than before, and therefore the packet is making progress toward its destination. (Otherwise we suspect malicious intervention and drop the packet.) If we can *guarantee that a packet is closer to its destination with every hop*, we can bound the potential damage from an attacker as a function of network size. (A more desirable property is to guarantee *good* progress, such as logarithmic path length, but both allow us to obtain an upper bound on attack success.)

PLGP *does not* satisfy no-backtracking. In non-source routing protocols, routes are dynamically composed of forwarding decisions made independently by each node. PLGP differs from other protocols in that packets paths are further bounded by a tree, forwarding packets along the shortest route through the tree that is allowed by the physical topology. In other words, packet paths are constrained both by physical neighbor relationships and the routing tree. Since the tree implicitly mirrors the topology (two nodes have the same parent if and only if they are physical neighbors, and two nodes sharing an ancestor have a network path to each other), and since every node holds an identical copy of the address tree, every node can verify the optimal next logical hop. However, this is not sufficient for no-backtracking to hold, since nodes cannot be certain of the path previously traversed by a packet. Communicating a local view of route cost is not as easy as it seems, since adversaries can always lie about their local metric, and so PLGP is still vulnerable to directional antenna/wormhole attacks, which allow adversaries to divert packets to any part of the network.

To preserve no-backtracking, we add a verifiable path history to every PLGP packet, similar to route authentications in Ariadne [29] and path-vector signatures in [70]. The resulting protocol, PLGP with attestations (PLGPa) uses this packet history together with PLGP's tree routing structure so every node can securely verify progress, *preventing any significant adversarial influence on the path taken by any packet which traverses at least one honest node*. Whenever node n forwards packet p, it this by attaching a non-replayable attestation (signature). These signatures form a chain attached to every packet, allowing any node receiving it to validate its path. Every forwarding node verifies the attestation chain to ensure that the packet has never travelled away from its destination in the logical address space. See Function `secure_forward_packet` for the modified protocol.

PLGPa satisfies no-backtracking. To show that our modified protocol preserves the no-backtracking property, we define a network as a collection of nodes, a topology, connectivity properties, and node identities, borrowing the model used by Poturalski *et al.* in [57]. Honest nodes can broadcast and receive messages, while malicious nodes can also use directional antennas to transmit to (or receive from) any node

in the network without being overheard by any other node. Honest nodes can compose, forward, accept, or drop messages, and malicious nodes can also arbitrarily transform them. Our adversary is assumed to control m nodes in an N-node network (with their corresponding identity certificates and other secret cryptographic material) and has perfect knowledge of the network topology. Finally, the adversary cannot affect connectivity between any two honest nodes.

Since all messages are signed by their originator, messages from honest nodes cannot be arbitrarily modified by malicious nodes wishing to remain undetected. Rather, the adversary can only alter packet fields that are changed en route (and so are not authenticated), so only the route attestation field can be altered, shortened, or removed entirely. To prevent truncation, which would allow Vampires to hide the fact that they are moving a packet away from its destination, we use Saxena and Soh's one-way signature chain construction [64], which allow nodes to add links to an existing signature chain, but not remove links, making attestations append-only.

For the purposes of Vampire attacks, we are unconcerned about packets with arbitrary hop counts that are never received by honest nodes but rather are routed between adversaries only, so we define the hop count of a packet as follows:

Definition 2. *The hop count of packet p, received or forwarded by an honest node, is no greater than the number of entries in p's route attestation field, plus 1.*

When any node receives a message, it checks that every node in the path attestation 1) has a corresponding entry in the signature chain, and 2) is logically closer to the destination than the previous hop in the chain (see Function `secure_forward_packet`). This way, forwarding nodes can enforce the forward progress of a message, preserving no-backtracking. If no attestation is present, the node checks to see if the originator of the message is a physical neighbor. Since messages are signed with the originator's key, malicious nodes cannot falsely claim to be the origin of a message, and therefore do not benefit by removing attestations.

Theorem 1. *A PLGPa packet p satisfies no-backtracking in the presence of an adversary controlling $m < N - 3$ nodes if p passes through at least one honest node.*

Proof:

Consider two arbitrary PLGPa protocol traces $\mathcal{H}$ and $\mathcal{M}$ of the same N-node network, in which node S sends packet p to node D. Constrain $\mathcal{H}$ such that all nodes are honest, and constrain $\mathcal{M}$ such that $m < N - 3$ are malicious. Let p reach an arbitrary honest node I along the protocol-defined packet path in h hops in $\mathcal{H}$, but in $h + \delta$ hops for $\delta > 0$ in $\mathcal{M}$ (no-backtracking is *not* satisfied in the latter). Since PLGPa is deterministic, the difference δ must be attributable to a malicious node. Further, since the hop count of p when it arrives at I is greater in $\mathcal{M}$ than in $\mathcal{H}$, p's route attestation chain must be δ longer in $\mathcal{M}$. Recall that every node has a unique virtual address, and no packet may be forwarded between any two nodes without moving either backward or forward through the virtual address space, so p must have moved backward in the coordinate space by at least one hop.[9]

Consider the following three scenarios: 1) I is a neighbor of S and the next hop of p; 2) I is a neighbor of D and the last hop of p before the destination; and 3) I is a forwarding node of the packet, but is neither a neighbor of S nor D. If I forwards a packet with $h + \delta$ hops in its route attestation, the adversary must have succeeded in at least one of the following:

- causing honest node I to forward p with non-null attestation, over a route that backtracked, violating the assumption that honest nodes correctly follow PLGPa;[10]
- causing honest node I to forward p with a non-null attestation, from source S who is I's direct neighbor, violating the assumption that honest nodes correctly follow PLGPa;
- truncating the route attestation, violating the security of chain signatures.

Finally, if I forwards p with a null attestation, it is either a neighbor of S or the adversary has broken the signature scheme used by the sender to attest to the packet's invariant fields — an honest I would not forward a packet with no attestation if the packet source is not a neighbor.[11] Since each possible adversarial action which results in backtracking violates an assumption, the proof is complete. ■

Since no-backtracking guarantees packet progress, and PLGPa preserves no-backtracking, it is the only protocol discussed so far that provably bounds *the ratio of energy used in the adversarial scenario to that used with only honest nodes to 1*, and by the definition of no-backtracking PLGPa resists Vampire attacks. This is achieved because packet progress is securely verifiable. Note that we cannot guarantee that a packet will reach its destination, since it can always be dropped. Guaranteed delivery is beyond the scope of this paper.

In strictly enforced no-backtracking, topology changes that may eliminate all protocol-level paths to a node that do not require backtracking, even though network-level paths still exist (e.g. the GPSR "dead end" scenario). To deal with such situations we can allow for limited backtracking (α-backtracking, as opposed to our original 0-backtracking scheme), which provides some leeway in the way no-backtracking is verified, allowing a certain amount of total backtracking per packet within the security parameter α. The extended security proof by induction on α is trivial.

VII. Performance considerations

PLGP imposes increased setup cost over BVR [21], but compares favorably to in terms of packet forwarding overhead. While path stretch increases by a factor of 1.5–2, message delivery success without resorting to localized flooding is improved: PLGP never floods, while BVR must flood 5–10% of packets depending on network size and topology [53]. PLGP also demonstrates more equitable routing load distribution and path diversity than BVR. Since the forwarding phase should

[9]Malicious nodes who choose to not update route attestations may pass p indefinitely between themselves, but this does not increase the hop count as defined (malicious nodes would only be draining their own battery).

[10]This case subsumes a malicious source and non-null packet attestation.

[11]A malicious source transmitting the packet with a null attestation is allowed without loss of generality.

last considerably longer than setup, PLGP offers performance comparable to BVR in the average case.

PLGPa includes path attestations, increasing the size of every packet, incurring penalties in terms of bandwidth use, and thus radio power. Adding extra packet verification requirements for intermediate nodes also increases processor utilization, requiring time and additional power. Of course there is nothing to be gained in completely non-adversarial environments, but in the presence of even a small number of malicious nodes, the increased overhead becomes worthwhile when considering the potential damage of Vampire attacks.

The bandwidth overhead of our attestation scheme is minimal, as chain signatures are compact (less than 30 bytes). Comparatively, a *minimum-size* DSR route request packet with no route, payload, or additional options is 12 bytes [35]; we used 512-byte data packets in our simulations. The additional bandwidth, therefore, is not significant, increasing per-packet transmit power by about $4.8\mu J$, plus roughly half for additional power required to receive [66].

Energy expenditure for cryptographic operations at intermediate hops is, unfortunately, much greater than transmit or receive overhead, and much more dependent on the specific chipset used to construct the sensor. However, we can make an educated guess about expected performance and power costs. Highly-optimized software-only implementations of AES-128, a common *symmetric* cryptographic primitive, require about 10 to 15 cycles per byte of data on modern 32-bit x86 processors without AES-specific instruction sets or cryptographic co-processors [6]. Due to the rapid growth in the mobile space and increased awareness of security requirements, there has been significant recent work in evaluating symmetric and asymmetric cryptographic performance on inexpensive and low-power devices. Bos *et al.* report AES-128 performance on 8-bit microcontrollers of 124.6 and 181.3 CPU cycles per byte [9], and Feldhofer *et al.*report just over 1000 cycles per byte using low-power custom circuits [20]. Surprisingly, although *asymmetric* cryptography is generally up to two orders of magnitude slower than symmetric, McLoone and Robshaw demonstrate a fast and low-power implementation of an asymmetric cryptosystem for use in RFID tags [47]. Their circuitry uses 400 to 800 cycles per round (on 8- and 16-bit architectures, respectively) in the high-current configuration (comparable in terms of clock cycles to AES for RFID [20], but with half to one-tenth the gates and vastly less power), and 1088 cycles when using about 6 times less current.

Chain signatures are a somewhat more exotic construction, and require *bilinear maps*, potentially requiring even more costly computation than other asymmetric cryptosystems. Bilinear maps introduce additional difficulties in estimating overhead due to the number of "pairings"from which implementers can choose. Kawahara *et al.* use Tate pairings, which are almost universally accepted as the most efficient [22], and show that their Java implementation has similar mobile phone performance as 1024-bit RSA [62] or 160-bit elliptic curve (ECC) [8] cryptosystems [38]. Scott *et al.* show that modern 32-bit smartcards can compute Tate pairings in as little as 150ms — comparable efficiency to symmetric cryptography [65]. Furthermore, English *et al.* show how to construct hardware to perform bilinear map operations in about 75,000 cycles at 50MHz (1.5ms) using $5.79\mu J$ [18].[12] When using specialized hardware for bilinear map computation, power requirements for chain signature-compatible cryptographic operations are roughly equivalent to for transmission of the 30-byte chain signature. Assuming a node performs both signature verification as well as a signature append operation, adding attestations to PLGP introduces roughly the same overhead as increasing packet sizes by 90 bytes, taking into account transmit power and cryptographic operations. Without specialized hardware, we estimate cryptographic computation overhead, and thus increased power utilization, of a factor of 2–4 per packet on 32-bit processors, but *mostly independent of the route length or the number of nodes in the network*: while the hop record and chain signature do grow, their size increase is negligible. In other words, the overhead is constant ($O(1)$) for a given network configuration (maximum path length), and cannot be influenced by an adversary. Fortunately, hardware cryptographic accelerators are increasingly common and inexpensive to compensate for increased security demands on low-power devices, which lead to increased computational load and reduced battery life [17, 18, 20, 33, 39, 46, 47, 49, 56].

In total, the overhead *on the entire network* of PLGPa (over PLGP) when using 32-bit processors *or* dedicated cryptographic accelerator is the energy equivalent of 90 additional bytes per packet, or a factor $O(x\lambda)$, where λ is the path length between source and destination and x is 1.2–7.5, depending on average packet size (512 and 12 bytes, respectively). Even without dedicated hardware, the cryptographic computation required for PLGPa is tractable even on 8-bit processors, although with up to a factor of 30 performance penalty, but this hardware configuration is increasingly uncommon.

VIII. Securing the discovery phase

Without fully solving the problem of malicious topology discovery, we can still mitigate it by forcing synchronous discovery and ignoring discovery messages during the intervening periods. This can lead to some nodes being separated from the network for a period of time, and is essentially a form of rate limiting. Although we rejected rate limiting before, it is acceptable here since discovery should consume a small fraction of running time compared to packet forwarding. We can enforce rate limits in a number of ways, such as neighbor throttling [35] or one-way hash chains [14]. We can also optimize discovery algorithms [32] to minimize our window of vulnerability. If a network survives the high-risk discovery period, it is unlikely to suffer serious damage from Vampires during normal packet forwarding.

While PLGPa is not vulnerable to Vampire attacks during the forwarding phase, we cannot make the same claim about discovery. However, we *can* give some intuition as to how to further modify PLGPa to bound damage from malicious discovery. (The value of that bound in practice remains an open problem.) The major issue is that malicious nodes can

[12]Even on 8-bit ATmega128 series microcontrollers commonly used in Mica sensor "motes" [27], Tate pairings can be computed in under 31 seconds [51], and elliptic curve operations take less than a second [17].

use directional antennas to masquerade neighbors to any or all nodes in the network, and therefore look like a group of size one, with which other groups will try to preferentially merge. Merge requests are composed of the requested group's ID as well as all the group members' IDs, and the receiving node will flood this request to other group members. Even assuming groups generate signed tokens that cost no energy to verify, a Vampire would be able to flood its group with every group descriptor it knows, and use its directional antenna to snoop on broadcasts outside their neighbor range, relaying merge requests from entirely honest groups. Since each Vampire will start as a group of one, other groups will issue merge requests, which the Vampire can deny. In PLGP, denials are only allowed if another merge is in progress, so if we modify the reject message to include the ID of the group with which the merge is in progress (and a signature for non-repudiation), these messages can be kept and replayed at the end of the topology discovery period, detecting and removing nodes who incorrectly deny merge requests. Therefore, Vampires reject legitimate merge requests at their own peril. Any group containing a Vampire can be made to serially join with a "group" composed only of each Vampire in the network (all of them would have to advertise themselves as neighbors of each group). Even wholly honest groups can be fooled using directional antennas: Vampires could maintain the illusion that it is a neighbor of a given group. Since join events require multiparty computation and are flooded throughout the group, this makes for a fairly effective attack. PLGP already provides for the discovery of such subterfuge upon termination of topology discovery: a node who is a member of multiple groups will be detected once those groups join (and all groups are guaranteed to merge by the end of the protocol).

Since PLGP offers the chance to detect active Vampires once the network converges, successive re-discovery periods become safer. This is more than can be said of other protocols, where malicious behavior during discovery may go undetected, or at least unpunished. However, the bound we can place on malicious discovery damage in PLGPa is still unknown. Moreover, if we can conclude that a single malicious node causes a factor of k energy increase during discovery (and is then expelled), it is not clear how that value scales under collusion among multiple malicious nodes.

IX. Conclusion

In this paper we defined Vampire attacks, a new class of resource consumption attacks that use routing protocols to permanently disable ad-hoc wireless sensor networks by depleting nodes' battery power. These attacks do not depend on particular protocols or implementations, but rather expose vulnerabilities in a number of popular protocol classes. We showed a number of proof-of-concept attacks against representative examples of existing routing protocols using a small number of weak adversaries, and measured their attack success on a randomly-generated topology of 30 nodes. Simulation results show that depending on the location of the adversary, network energy expenditure during the forwarding phase increases from between 50 to 1,000 percent. Theoretical worst-case energy usage can increase by as much as a factor of $O(N)$ per adversary per packet, where N is the network size. We proposed defenses against some of the forwarding-phase attacks and described PLGPa, the first sensor network routing protocol that provably bounds damage from Vampire attacks by verifying that packets consistently make progress toward their destinations. We have not offered a fully satisfactory solution for Vampire attacks during the topology discovery phase, but suggested some intuition about damage limitations possible with further modifications to PLGPa. Derivation of damage bounds and defenses for topology discovery, as well as handling mobile networks, is left for future work.

Acknowledgements

We would like to thank Hal Peterson, Tian He, Yongdae Kim, Daniel Andresen, and our anonymous reviewers for their very helpful comments on earlier drafts of this paper.

References

[1] *The network simulator — ns-2*. http://www.isi.edu/nsnam/ns/.
[2] Imad Aad, Jean-Pierre Hubaux, and Edward W. Knightly, *Denial of service resilience in ad hoc networks*, MobiCom, 2004.
[3] Gergely Acs, Levente Buttyan, and Istvan Vajda, *Provably secure on-demand source routing in mobile ad hoc networks*, IEEE Transactions on Mobile Computing **05** (2006), no. 11.
[4] Tuomas Aura, *Dos-resistant authentication with client puzzles*, International workshop on security protocols, 2001.
[5] John Bellardo and Stefan Savage, *802.11 denial-of-service attacks: real vulnerabilities and practical solutions*, USENIX security, 2003.
[6] Daniel Bernstein and Peter Schwabe, *New AES software speed records*, INDOCRYPT, 2008.
[7] Daniel J. Bernstein, *Syn cookies*, 1996. http://cr.yp.to/syncookies.html.
[8] I.F. Blake, G. Seroussi, and N.P. Smart, *Elliptic curves in cryptography*, Vol. 265, Cambridge University Press, 1999.
[9] Joppe W. Bos, Dag Arne Osvik, and Deian Stefan, *Fast implementations of AES on various platforms*, 2009.
[10] Haowen Chan and Adrian Perrig, *Security and privacy in sensor networks*, Computer **36** (2003), no. 10.
[11] Jae-Hwan Chang and Leandros Tassiulas, *Maximum lifetime routing in wireless sensor networks*, IEEE/ACM Transactions on Networking **12** (2004), no. 4.
[12] Thomas H. Clausen and Philippe Jacquet, *Optimized link state routing protocol (OLSR)*, 2003.
[13] Jing Deng, Richard Han, and Shivakant Mishra, *Defending against path-based DoS attacks in wireless sensor networks*, ACM workshop on security of ad hoc and sensor networks, 2005.
[14] ———, *INSENS: Intrusion-tolerant routing for wireless sensor networks*, Computer Communications **29** (2006), no. 2.
[15] Sheetalkumar Doshi, Shweta Bhandare, and Timothy X. Brown, *An on-demand minimum energy routing protocol for a wireless ad hoc network*, ACM SIGMOBILE Mobile Computing and Communications Review **6** (2002), no. 3.
[16] John R. Douceur, *The Sybil attack*, International workshop on peer-to-peer systems, 2002.
[17] Hans Eberle, Arvinderpal Wander, Nils Gura, Sheueling Chang-Shantz, and Vipul Gupta, *Architectural extensions for elliptic curve cryptography over GF(2^m) on 8-bit microprocessors*, ASAP, 2005.
[18] T. English, M. Keller, Ka Lok Man, E. Popovici, M. Schellekens, and W. Marnane, *A low-power pairing-based cryptographic accelerator for embedded security applications*, SOCC, 2009.
[19] Laura M. Feeney, *An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks*, Mobile Networks and Applications **6** (2001), no. 3.
[20] Martin Feldhofer, Sandra Dominikus, and Johannes Wolkerstorfer, *Strong authentication for RFID systems using the AES algorithm*, CHES, 2004.
[21] Rodrigo Fonseca, Sylvia Ratnasamy, Jerry Zhao, Cheng T. Ee, David Culler, Scott Shenker, and Ion Stoica, *Beacon vector routing: Scalable point-to-point routing in wireless sensornets*, NSDI, 2005.

[22] Steven Galbraith, Keith Harrison, and David Soldera, *Implementing the tate pairing*, Algorithmic number theory, 2002.

[23] Sharon Goldberg, David Xiao, Eran Tromer, Boaz Barak, and Jennifer Rexford, *Path-quality monitoring in the presence of adversaries*, SIGMETRICS, 2008.

[24] Andrea J. Goldsmith and Stephen B. Wicker, *Design challenges for energy-constrained ad hoc wireless networks*, IEEE Wireless Communications **9** (2002), no. 4.

[25] R. Govindan and A. Reddy, *An analysis of internet inter-domain topology and route stability*, INFOCOM, 1997.

[26] Mina Guirguis, Azer Bestavros, Ibrahim Matta, and Yuting Zhang, *Reduction of quality (RoQ) attacks on Internet end-systems*, INFOCOM, 2005.

[27] J.L. Hill and D.E. Culler, *Mica: a wireless platform for deeply embedded networks*, IEEE Micro **22** (2002), no. 6.

[28] Yih-Chun Hu, David B. Johnson, and Adrian Perrig, *SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks*, IEEE workshop on mobile computing systems and applications, 2002.

[29] Yih-Chun Hu, Adrian Perrig, and David B. Johnson, *Ariadne: A secure on-demand routing protocol for ad hoc networks*, MobiCom, 2002.

[30] ———, *Packet leashes: A defense against wormhole attacks in wireless ad hoc networks*, INFOCOM, 2003.

[31] ———, *Rushing attacks and defense in wireless ad hoc network routing protocols*, WiSE, 2003.

[32] Yangcheng Huang and Saleem Bhatti, *Fast-converging distance vector routing for wireless mesh networks*, ICDCS, 2008.

[33] D. Hwang, Bo-Cheng Lai, P. Schaumont, K. Sakiyama, Yi Fan, Shenglin Yang, A. Hodjat, and I. Verbauwhede, *Design flow for HW/SW acceleration transparency in the thumbpod secure embedded system*, Design automation conference, 2003.

[34] L. Iannone, R. Khalili, K. Salamatian, and S. Fdida, *Cross-layer routing in wireless mesh networks*, International symposium on wireless communication systems, 2004.

[35] David B. Johnson, David A. Maltz, and Josh Broch, *DSR: the dynamic source routing protocol for multihop wireless ad hoc networks*, Ad hoc networking, 2001.

[36] Chris Karlof and David Wagner, *Secure routing in wireless sensor networks: attacks and countermeasures*, IEEE international workshop on sensor network protocols and applications, 2003.

[37] Brad Karp and H.T. Kung, *GPSR: Greedy perimeter stateless routing for wireless networks*, MobiCom, 2000.

[38] Y. Kawahara, T. Takagi, and E. Okamoto, *Efficient implementation of Tate pairing on a mobile phone using Java*, International conference on computational intelligence and security, 2006.

[39] Manuel Koschuch, Joachim Lechner, Andreas Weitzer, Johann Groschdl, Alexander Szekely, Stefan Tillich, and Johannes Wolkerstorfer, *Hardware/software co-design of elliptic curve cryptography on an 8051 microcontroller*, CHES, 2006.

[40] Alexander Kröller, Sándor P. Fekete, Dennis Pfisterer, and Stefan Fischer, *Deterministic boundary recognition and topology extraction for large sensor networks*, Annual ACM-SIAM symposium on discrete algorithms, 2006.

[41] Aleksandar Kuzmanovic and Edward W. Knightly, *Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants*, SIGCOMM, 2003.

[42] Yu-Kwong Kwok, Rohit Tripathi, Yu Chen, and Kai Hwang, *HAWK: Halting anomalies with weighted choking to rescue well-behaved TCP sessions from shrew DDoS attacks*, Networking and mobile computing, 2005.

[43] Xiaojun Lin, N.B. Shroff, and R. Srikant, *A tutorial on cross-layer optimization in wireless networks*, Selected Areas in Communications, IEEE Journal on **24** (2006), no. 8.

[44] Xiapu Luo and Rocky K. C. Chang, *On a new class of pulsing denial-of-service attacks and the defense*, NDSS, 2005.

[45] Morteza Maleki, Karthik Dantu, and Massoud Pedram, *Power-aware source routing protocol for mobile ad hoc networks*, ISLPED, 2002.

[46] Yusuke Matsuoka, Patrick Schaumont, Kris Tiri, and Ingrid Verbauwhede, *Java cryptography on kvm and its performance and security optimization using hw/sw co-design techniques*, CASES, 2004.

[47] M. McLoone and M. Robshaw, *Public key cryptography and RFID tags*, CT-RSA, 2006.

[48] Timothy J. McNevin, Jung-Min Park, and Randolph Marchany, *pTCP: A client puzzle protocol for defending against resource exhaustion denial of service attacks*, Technical Report TR-ECE-04-10, Department of Electrical and Computer Engineering, Virginia Tech, 2004.

[49] V.P. Nambiar, M. Khalil-Hani, and M.M.A. Zabidi, *Accelerating the AES encryption function in OpenSSL for embedded systems*, ICED, 2008.

[50] Asis Nasipuri and Samir R. Das, *On-demand multipath routing for mobile ad hoc networks*, International conference on computer communications and networks, 1999.

[51] L.B. Oliveira, D.F. Aranha, E. Morais, F. Daguano, J. Lopez, and R. Dahab, *TinyTate: Computing the Tate pairing in resource-constrained sensor nodes*, NCA, 2007.

[52] Kihong Park and Heejo Lee, *On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack*, INFOCOM, 2001.

[53] Bryan Parno, Mark Luk, Evan Gaustad, and Adrian Perrig, *Secure sensor network routing: A clean-slate approach*, CoNEXT, 2006.

[54] Vern Paxson, *An analysis of using reflectors for distributed denial-of-service attacks*, SIGCOMM Comput. Commun. Rev. **31** (2001), no. 3.

[55] Charles E. Perkins and Pravin Bhagwat, *Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers*, Conference on communications architectures, protocols and applications, 1994.

[56] R. Potlapally, S. Ravi, A. Raghunathan, R.B. Lee, and N.K. Jha, *Impact of configurability and extensibility on IPSec protocol execution on embedded processors*, International conference on VLSI design, 2006.

[57] Marcin Poturalski, Panagiotis Papadimitratos, and Jean-Pierre Hubaux, *Secure neighbor discovery in wireless networks: Formal investigation of possibility*, ACM ASIACCS, 2008.

[58] Daniele Raffo, Cédric Adjih, Thomas Clausen, and Paul Mühlethaler, *An advanced signature system for OLSR*, SASN, 2004.

[59] David R. Raymond, Randy C. Marchany, Michael I. Brownfield, and Scott F. Midkiff, *Effects of denial-of-sleep attacks on wireless sensor network MAC protocols*, IEEE Transactions on Vehicular Technology **58** (2009), no. 1.

[60] David R. Raymond and Scott F. Midkiff, *Denial-of-service in wireless sensor networks: Attacks and defenses*, IEEE Pervasive Computing **7** (2008), no. 1.

[61] Jennifer Rexford, Jia Wang, Zhen Xiao, and Yin Zhang, *BGP routing stability of popular destinations*, IMW, 2002.

[62] R. L. Rivest, A. Shamir, and L. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Communications of the ACM **21** (1978), no. 2.

[63] Volkan Rodoplu and Teresa H. Meng, *Minimum energy mobile wireless networks*, IEEE Journal on Selected Areas in Communications **17** (1999), no. 8.

[64] Amitabh Saxena and Ben Soh, *One-way signature chaining: a new paradigm for group cryptosystems*, International Journal of Information and Computer Security **2** (2008), no. 3.

[65] Michael Scott, Neil Costigan, and Wesam Abdulwahab, *Implementing cryptographic pairings on smartcards*, CHES, 2006.

[66] Rahul C. Shah and Jan M. Rabaey, *Energy aware routing for low energy ad hoc sensor networks*, WCNC, 2002.

[67] Suresh Singh, Mike Woo, and C. S. Raghavendra, *Power-aware routing in mobile ad hoc networks*, MobiCom, 1998.

[68] Frank Stajano and Ross Anderson, *The resurrecting duckling: security issues for ad-hoc wireless networks*, International workshop on security protocols, 1999.

[69] Ivan Stojmenovic and Xu Lin, *Power-aware localized routing in wireless networks*, IEEE Transactions on Parallel and Distributed Systems **12** (2001), no. 11.

[70] Lakshminarayanan Subramanian, Randy H. Katz, Volker Roth, Scott Shenker, and Ion Stoica, *Reliable broadcast in unknown fixed-identity networks*, Annual ACM SIGACT-SIGOPS symposium on principles of distributed computing, 2005.

[71] Haibin Sun, John C. S. Lui, and David K. Y. Yau, *Defending against low-rate TCP attacks: dynamic detection and protection*, ICNP, 2004.

[72] Curtis Villamizar, Ravi Chandra, and Ramesh Govindan, *BGP route flap damping*, 1998.

[73] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford, *CAPTCHA: Using hard AI problems for security*, Eurocrypt, 2003.

[74] Yue Wang, Jie Gao, and Joseph S.B. Mitchell, *Boundary recognition in sensor networks by topological methods*, Annual international conference on mobile computing and networking, 2006.

[75] Anthony D. Wood and John A. Stankovic, *Denial of service in sensor networks*, Computer **35** (2002), no. 10.

[76] Guang Yang, M. Gerla, and M.Y. Sanadidi, *Defense against low-rate TCP-targeted denial-of-service attacks*, ISCC, 2004.

[77] Jun Yuan, Zongpeng Li, Wei Yu, and Baochun Li, *A cross-layer optimization framework for multihop multicast in wireless mesh networks*, IEEE Journal on Selected Areas in Communications **24** (2006), no. 11.

[78] Manel Guerrero Zapata and N. Asokan, *Securing ad hoc routing protocols*, WiSE, 2002.

Eugene Vasserman is an Assistant Professor of Computing and Information Sciences at Kansas State University. He received his Ph.D. and Master's degrees in Computer Science in 2010 and 2008, respectively, University of Minnesota. His B.S. in Biochemistry and Neuroscience with a Computer Science minor, is also from the University of Minnesota (2003). He is interested in distributed network security, privacy and anonymity, low-power and pervasive computing, peer-to-peer systems, and applied cryptography.

Nicholas Hopper is an Associate Professor in the department of Computer Science and Engineering, University of Minnesota. He received a B.A. with majors in Mathematics and Computer Science from the University of Minnesota-Morris in 1999 and a PhD in Computer Science from Carnegie Mellon University in 2004. He received the NSF CAREER award in 2006 and the Institute of Technology Student Board "Professor of the Year" award in 2007 and the McKnight Land-Grant award in 2008. His research interests include cryptography, anonymity, and distributed systems security.