This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2010 proceedings
This paper was presented as part of the main Technical Program at IEEE INFOCOM 2010.

Castor: Scalable Secure Routing for Ad Hoc Networks

Wojciech Galuba, Panos Papadimitratos, Marcin Poturalski, Karl Aberer
Ecole Polytechnique Fédérale de Lausanne (EPFL)
firstname.lastname@epfl.ch

Zoran Despotovic, Wolfgang Kellerer
DOCOMO Euro-Labs, Munich, Germany
lastname@docomolab-euro.ch

***Abstract*—Wireless ad hoc networks are inherently vulnerable, as any node can disrupt the communication of potentially any other node in the network. Many solutions to this problem have been proposed. In this paper, we take a fresh and comprehensive approach that addresses simultaneously three aspects: security, scalability and adaptability to changing network conditions. Our communication protocol, Castor, occupies a unique point in the design space: It does not use any control messages except simple packet acknowledgements, and each node makes routing decisions locally and independently without exchanging any routing state with other nodes. Its novel design makes Castor resilient to a wide range of attacks and allows the protocol to scale to large network sizes and to remain efficient under high mobility. We compare Castor against four representative protocols from the literature. Our protocol achieves up to two times higher packet delivery rates, particularly in large and highly volatile networks, while incurring no or only limited additional overhead. At the same time, Castor is able to survive more severe attacks and recovers from them faster.**

I. Introduction

Ad hoc networks pose a significant security challenge: the adversary could degrade or even deny communication by abusing their self-organizing operation. *Secure communication* protocols should be able to maintain acceptable data delivery rates under all feasible attacks. *Secure route discovery* can ensure that data are not sent across routes manipulated by the adversary so that they never lead to the sought destination. But adversaries could still disrupt communication (e.g., drop or corrupt packets) once securely discovered routes are being used.

It is thus necessary, as some protocols surveyed in §II do, to utilize a *secure data transmission* protocol on top of secure route discovery. Secure data transmission protocols correlate data delivery failures with specific routes or network areas, possibly controlled by the adversary. Then, they reroute the traffic, to avoid the adversary and reestablish reliable communication.

This approach was shown to be effective, especially if sufficiently rich connectivity information is available. Simultaneous use of multiple paths, redundancy in transmissions, and end-to-end secure feedback allow for quick route convergence [1]. However, because of system constraints, there may not be enough bandwidth available for multi-path data transmission. When resources are scarce, the generally applicable solution is a single-path secure communication protocol: sending data and feedback across a single path, and switching to another path once the current one is deemed unreliable.

Would such a solution remain efficient and effective in large and highly volatile networks, even in the presence of powerful adversaries? Consider networks that are open, mobile and can grow in size, with the subset of nodes supporting a given single-path flow constantly changing: managing the variability, avoiding the faulty and adversarial nodes, while sustaining reliable communication is a challenge.

Our *Continuously Adapting Secure Topology-Oblivious Routing (Castor)* addresses exactly this problem. Each node keeps track of the reliability of only its neighbors; none of the local state is ever exchanged over the network; packet sizes do not carry routes, thus their lengths do not grow with the network size; each node operates fully autonomously, making routing decisions independently of other nodes and without knowing the network topology beyond its local one-hop connectivity. These features basically make Castor *scalable*. Moreover, in-network routing state allows Castor to *rapidly adapt* to a wide range of faults, malicious and benign, even under an *overwhelming adversarial presence*. Finally, the minimal exchange of information between the nodes implies there is little need for authenticating it or securing its transmission; this enables Castor to operate under the *simplest*, among those in the literature, *trust assumptions*.

Our extensive comparative evaluation shows that Castor outperforms four other protocols (SRP/SSP, Sprout, SEAD, and trivially the non-secure AODV), with significant advantages: (i) Castor consistently achieves up to a *40% higher packet delivery rate without any additional overhead*, (ii) it recovers at least *twice as fast* as the other protocols, (iii) it is the *only one* among the five evaluated here that achieves *full recovery* from the wormhole attack without the help of a secure neighbor discovery protocol. Equally important, Castor maintains its advantage as the network *scales* and mobility increases: For example, in a 400 node network with 80 black-holes and continuous mobility, Castor achieves, with a mild overhead increase, a consistent 60% packet delivery rate, *double* that of other protocols.

In summary, our main **contributions** in this paper are: (a) Castor, a scalable secure communication protocol that is highly resilient to a wide range of attacks and benign faults, and (b) an extensive comparative performance evaluation with four other protocols, under various attacks, mobility and network size settings. What sets Castor apart is its fundamentally novel approach, among secure communication protocols, and its versatility, in spite its simplicity. Castor is the first protocol to demonstrate robustness against such an large spectrum of attacks and for a wide range of network scales. Our comprehensive comparative evaluation of secure communication protocols is also the first of its kind.

In the rest of the paper, we first discuss related work (§II) and give an overview of Castor. Then, we define the system and adversary models (§IV) and present in detail the functionality

978-1-4244-5837-0/10/$26.00 ©2010 IEEE

of Castor (§V). We analyze its security (§VI) and evaluate the performance (§VII) before we conclude.

II. Related Work

Secure ad hoc networking protocols address two main issues: (a) secure route discovery, to prevent attacks on the disseminated routing information, and (b) secure data transmission, to ensure data delivery. Most proposals in the literature consider the first issue only or assume the second one is addressed by upper layer protocols; few consider both issues.

Secure route discovery. SRP [2] is an on-demand protocol: it floods in a controlled manner a route request (RREQ), with intermediate nodes each appending its identifier. The destination returns route reply (RREP) packets strictly across the reverse of the path accumulated in the RREQs. End-nodes can authenticate each other and their RREQ and RREP packets; intermediate nodes do not need to authenticate traffic from end-nodes. Ariadne [3] follows the same principle, but it authenticates intermediate nodes at the end-nodes. This increases the trust management complexity and overhead, in return for stricter identification of the intermediate nodes at the source. endairA [4] takes essentially the same approach, utilizing only public key cryptography, and offers increased resilience to attacks.

SRP, Ariadne, and endairA provide the entire discovered route (connectivity information) to the source node, and the same is true for link state protocols such as SLSP [5]. In a different category, implicit route discovery protocols [6] provide each node with the next hop towards the destination: ARAN [7] discovers a single route, based on the first-returning RREP at the source; S-AODV [8] provides security for AODV [9], authenticating its RREQ, RREP and route error packets; and SEAD [10] protects distance-vector calculations from distance decrease, using symmetric key cryptography (whereas ARAN and S-AODV use digital signatures).

Secure data transmission. SSP [11] is a secure single-path protocol that relies on an end-to-end security association; it transmits packets across a route calculated over the connectivity the underlying route discovery provides (typically, protocols such as SRP). The destination validates received data and responds with acknowledgements; if not, the source detects a packet loss. The route rating is increased each time an acknowledgement is received, and it is reduced when a timeout occurs (no ACK); once the rating drops below a threshold, the route is discarded and the source switches to another one (invoking a new route discovery if needed). SSP is robust to any attack (e.g., wormholes, tunnels, other collusion attacks) that causes a packet to be dropped; if so, the route is discarded.

Sprout [12] is a protocol that source-routes data across a single path chosen among many alternative ones. These paths are calculated over the topology view that a secure link state discovery protocol offers, with nodes broadcasting link state updates across the network. In order to be resilient against colluding adversaries that advertise fictitious links, Sprout introduces mechanisms that prevent the pollution of the network link state view. Routes are generated and utilized probabilistically, acknowledgements are returned by the destination, and routes deemed operational are re-used and new alternative ones are explored. The link-state operation requires that any node can identify all other nodes at all times. SSP and Sprout are the two protocols closer to our Castor.

Other related schemes. ODSBR[13] discovers routes reactively, it updates link weights based on their behavior observed at the sources, and when communication reliability drops below a threshold, it augments data packets with probes to identify the wrong-doer. ODSBR requires that the source knows all nodes in the network. It can maintain reliability across a route above a threshold unless two or more colluding attackers are part of the route [12]. Beyond security protocols, reputation and remuneration-based schemes have been considered [14]. All these schemes are complex and costly (e.g., requiring a full trust graph, long observation periods), or they are effective only against rational adversaries, or they are susceptible to attacks that incriminate correct nodes. Finally, a note on the so-called ant-based routing protocols [15], [16], [17] that somewhat resemble Castor: they do not have an explicit route discovery phase and ACKs reinforce paths (by analogy to phermone traces). However, none of them considers security.

Comparison to Castor. In brief, Castor extends over secure route discovery, as it falls in the category of the comprehensive solutions that also secure data transmission; e.g., ODSBR, SRP plus SSP or SMT, and Sprout. Compared to these, it introduces significant differences. Concisely put: (i) Routes need not be attached to packets, thus packets do not grow in length with the network size (and thus route length), (ii) there are no route discovery control packets, only data and acknowledgements, (iii) the communication reliability information is kept locally at each node in the network, not at the source, (iv) Castor does not seek to identify and exclude attackers, and (v) it relies on the simplest trust management assumptions (same as those of the SRP-SSP combination).

III. Protocol design overview

Castor operates as follows: When the source sends a packet, intermediate nodes forward it until it reaches its destination, which then responds with an acknowledgement that follows the reverse path back to the source.

Learning from failures. Nodes locally keep per-flow-and-next-hop reliability metrics (§V-D), which are updated constantly, based on the arriving acknowledgements indicating success and the acknowledgement timeouts indicating failure. These metrics are used to select the most reliable next hop for each incoming packet. If no reliable next hop exists, or if the recorded history is insufficient for the node to make a choice, the packet is locally broadcasted. Each node decides whether to unicast or broadcast independently (§V-C).

Reliability as a primary metric. Protocols that minimize the number of hops or the round-trip time, are susceptible to an attacker advertising shorter routes or setting up wormholes or tunnels to attract traffic and then drop passing data packets. To be robust against such attacks, Castor uses reliability as its primary metric. As the routing is reliability-driven, Castor is able to detect and react to all causes of packet loss, independent of their nature, be it benign or adversarial.

Response time as a secondary metric. For performance reasons, Castor keeps routes short by giving preference to the first neighbor responding with an acknowledgment during the route discovery. However, this neighbor can be routed around if it turns out to be unreliable.

Emergent secure routes. With every node estimating and acting on local reliability metrics, Castor is able to locally route to avoid unreliable neighbors. This results in fast global convergence to reliable routes and efficient continuous adaptation under mobility.

Local repair. In contrast to some other protocols, adversarial or benign failures do not cause costly network-wide floods to search for better routes. Most of the time, Castor has another reliable next hop to switch to. It resorts to broadcasting only when it faces severe failures. In most cases, a brief cascade of local broadcasts reaches a part of the network with reliable next hops and unicasting resumes.

Secure, isolated routing state. Nodes make routing decisions independently, based only on locally accumulated neighbor reliability metrics. In other words, nodes are oblivious to any network connectivity information beyond the local neighbors. No routing state is ever exchanged between nodes, which removes the problem of securing the information exchange. State locality and minimal control traffic are also key to Castor's scalability (§VII-D).

Routing state is stored on a per-flow, not per-destination, basis. A cryptographic scheme ensures that only the packets coming from the flow's source and the acknowledgments coming from the flow's destination can influence the routing state for that flow (§V-B). Despite relying on simple trust assumptions, these mechanisms provide a strong protection against routing state pollution by the adversary.

IV. Assumptions

A. System Model

We consider a wireless ad hoc network composed of static or mobile *nodes*, i.e., computing platforms with wireless transceivers, with limited communication range. Nodes communicate directly over the wireless channel with their *neighbors*. Nodes assist other nodes with communication across multiple links (hops). Each node has a unique identity that can be cryptographically validated if needed. Nodes that conform to system protocols are *correct*, and those that deviate from them are *adversarial*.

Cryptography. Castor requires that for each pair of end nodes, a source s and a destination d, that wish to communicate securely across the network, either s and d share a pre-established symmetric key $K_{s,d}$ or s knows the public key K_d of d. Further, we assume d is able to verify the integrity of the messages sent by s. We also assume that any two correct neighboring nodes can establish a shared secret symmetric key to authenticate their communication. Neighbor-to-neighbor keys can be established and authenticated in a number of ways, depending on the system instantiation, e.g., through key transport or agreement. Authentication can be performed with the help of local channels, passwords, certificates etc. For example, a certified public key can serve as the verifiable unique identity of a node. Moreover, each correct node can authenticate messages it broadcasts at the data link layer, by utilizing symmetric-key schemes such as [18].

Neighbor discovery. Neighbors are discovered by a simple mechanism, such as beaconing. We do *not* require a secure neighbor discovery (SND) protocol, which would prevent the adversary from convincing two non-neighbor nodes that they are neighbors.

B. Adversary Model

The adversary controls a number of adversarial nodes, which can be internal or external. The internal nodes are equipped with the same cryptographic material as the correct nodes. For example, a compromised but previously correct node can become an internal adversary. A single adversarial node can appear as multiple network nodes by utilizing the compromised identities and cryptographic keys.

An adversarial node can arbitrarily deviate from the protocol definition. In particular, it can drop, modify, and replay any message. The adversary is, however, computationally bounded and cannot break cryptographic primitives. If beneficial, an adversarial node can abide by the protocol for any period of time. Adversarial nodes can also act in coordination and mount collusion attacks. Moreover, adversarial nodes can communicate across large distances using fast out-of-band communication links (typically used to mount e.g., *wormhole* and *tunnel* attacks) and jam communication.

The objective of the adversary we consider here is denial of service: to prevent communication or, in other words, to prevent messages from being delivered. We do not seek to thwart any adversarial behavior that does not result in packet loss. In particular, we do not address the problem of preventing traffic interception, eavesdropping or analysis.

C. Metrics

We focus exclusively on flows between correct source-destination pairs. The primary performance metric is the *packet delivery rate* (PDR). More precisely, we are interested in the *network-layer PDR*. We want to capture the raw network performance in the presence of adversaries, without using any packet retransmission schemes, either at network or upper layers. Further, we are interested in the *bandwidth utilization* per delivered packet.

V. The protocol

A. Message Specification

Castor uses two types of messages: PKTs and ACKs. The payload packet **PKT** is a tuple $(s, d, H, b_k, f_k, e_k, M)$: s and d are the source/destination identfiers; H is the *flow identifier (id)*; b_k is the *PKT id*; f_k is the *flow authenticator*, used for verifying that the PKT belongs to flow H; e_k is an *encrypted ACK authenticator*. Finally, M is the payload, which typically includes an integrity protection mechanism.

The **ACK** has only one field a_k, an *ACK authenticator*, which is used for verifying that the corresponding PKT has been delivered to the destination.

B. Cryptographic Mechanisms

To ensure the correct routing state updates, the fields of the PKTs and ACKs need to satisfy two properties. First, an ACK a_k should only be received by an intermediate node if the destination has indeed received the corresponding PKT b_k. Second, no node except the source of flow H should be able to generate a PKT b_k that would be verified as belonging to H. We emphasize that we *do not* require an intermediate node to verify that all PKTs originate from some particular source – which is impossible as in our setup the source does not pre-share keys with intermediate nodes. Rather, an intermediate node verifies that all PKTs originate at the same, but arbitrary, source.

There are many cryptographic schemes that can achieve the desired properties. We present here an overview of an efficient solution based on Merkle hash trees; for a more detailed presentation (including the protocol pseudocode), as well as an alternative public-key scheme we refer the reader to [19].

PKT generation. For each flow that the source s wants to send to a destination d, the source pre-generates: (i) a set of random nonces, $a_1, \ldots, a_w$, the ACK authenticators, (ii) a corresponding set of PKT ids $b_k = h(a_k)$, where h is a cryptographic hash function and (iii) a Merkle hash tree with $h(b_1), \ldots, h(b_w)$ as leaves. The root of this tree becomes the flow-id H. The pre-generated values are then used when sending the k-th PKT $(s, d, H, b_k, f_k = [x_1, \ldots, x_l], e_k = E_{K_{sd}}(a_k), M)$; a_k is encrypted using the key K_{sd} shared between s and d. The integers $x_1, \ldots, x_l$ form a sequence of siblings of the vertices on the tree path from $h(b_k)$ up to H.

PKT verification. To verify that a PKT $(s, d, H, b_k, f_k = [x_1, \ldots, x_l], e_k, M)$ belongs to flow H, an intermediate node checks whether $h(\ldots h(h(h(b_k)||x_1)||x_2)|| \ldots x_l) = H$, i.e., if $h(b_k)$ is a leaf of the Merkle tree with root H. The $h(\ldots h(h(b_k||x_1)||x_2)|| \ldots x_l)$ is a shorthand notation, in practice the order of concatenations depends on the position of b_k in the Merkle tree. If the above check is successful, the PKT is forwarded and b_k is stored. Otherwise, the PKT is dropped. Note that unforgeabilty of b_k/f_k follows from the hardness of inverting the hash function h.

PKT verification at destination. In addition to the Merkle tree test, the destination performs additional verification of the PKT. First, it checks whether $b_k = h(D_{K_{sd}}(e_k))$. Then, it checks the integrity of the payload M. If all tests are successful, d accepts the PKT, and sends the corresponding ACK a_k to the neighbor who delivered the PKT. Otherwise, the PKT is dropped. Note that only the destination is able to generate a correct ACK without breaking the encryption of e_k or finding the pre-image of b_k under h.

ACK verification. Upon receiving and ACK a_k, a node computes $h(a_k)$ and checks whether it corresponds to any stored b_k. If yes, the ACK is accepted and rebroadcasted, and the routing state of the corresponding flow H is appropriately updated. Otherwise, the ACK is ignored.

C. PKT Forwarding

Basic forwarding. For every neighbor $j = 1 \ldots n$ and for every encountered flow H, a node i stores a *reliability estimator* $s_{H,j} \in [0, 1]$. Consider what happens when i either (1) receives a PKT, and verifies that it belongs to some flow H (§V-B) or (2) i is the source of the PKT. First, i attempts to forward the PKT to the most reliable neighbor, according to the values of all the reliability estimators for the flow H. If no neighbor is deemed reliable, the PKT is broadcasted to all the neighbors in search of more reliable routes. Immediately after the PKT is sent, i starts a timer T_{H,b_k} that times out after T_{ACK} if the corresponding ACK is not received.

The decision to unicast or broadcast is probabilistic. Let $p_{max} = \max_{j=1\ldots n} s_{H,j}$ be the value of the highest reliability estimator for the flow H among all the neighbors $j = 1 \ldots n$ of i. The probability that the PKT is broadcasted is $e^{-\gamma p_{max}}$, otherwise it is unicasted to the next hop with the highest reliability estimator. Ties are broken by choosing uniformly at random. The $\gamma > 0$ parameter allows for controlling the bandwidth investment in route discovery depending on the desired packet delivery rates (PDR).

Duplicate PKTs. If a node receives a PKT that it has received before, this PKT is not forwarded again. However, if an ACK corresponding to this PKT was received, the node rebroadcasts the ACK. If an intermediate node receives a PKT with b_k identical to some previously seen PKT, but with a *different* payload or encrypted ACK authenticator e_k, this PKT is forwarded further. This is because an intermediate node cannot tell which of the PKTs with the same authenticator are incorrect or forged. We explain this in more detail in §VI-B. The T_{H,b_k} timer is not restarted on PKT duplicates.

D. Updating the Reliability Estimators

Reliability estimators. The reliability estimator $s_{H,j}$ is an arithmetic average of two reliability estimators $s^a_{H,j}$ and $s^f_{H,j}$. Both reliability estimators are exponential averages of packet delivery rates. More precisely, let $\alpha^a_{H,j}$ be the running average of successful deliveries and $\beta^a_{H,j}$ the running average of failed deliveries; then $s^a_{H,j} = \frac{\alpha^a_{H,j}}{\alpha^a_{H,j}+\beta^a_{H,j}}$. Upon a failure, the updates are: $\alpha^a_{H,j} \leftarrow \delta\alpha^a_{H,j}$ and $\beta^a_{H,j} \leftarrow \delta\beta^a_{H,j} + 1$. We denote this negative update as $s^a_{H,j} \downarrow$. Upon success, the reliability estimator is positively ($s^a_{H,j} \uparrow$) updated: $\alpha^a_{H,j} \leftarrow \delta\alpha^a_{H,j} + 1$ and $\beta^a_{H,j} \leftarrow \delta\beta^a_{H,j}$. Initially, $\alpha^a_{H,j} = 0$ and $\beta^a_{H,j} = 1$. Updating of $s^f_{H,j}$ is analogous. The $0 < \delta < 1$ parameter controls how fast Castor adapts; the lower the value the faster the adaptation.

The $s^a_{H,j}$ estimator is updated more frequently than $s^f_{H,j}$ as we explain next. The "a" stands for "all ACKs" and "f" stands for "first ACK".

ACK timeout. Consider the case when T_{H,b_k} times out before the corresponding $ACK(a_k)$ is received. The update of the estimators then depends on whether the corresponding PKT was broadcasted or unicasted to some neighbor j. In the former case, no reliability estimators are changed. In the latter case, both $s^a_{H,j}$ and $s^f_{H,j}$ are decreased.

For simplicity, we are using a T_{ACK} timeout that is fixed. The timeout value should be set based on the delay conditions in the target network. A dynamically adapting ACK timeout similar to that of Sprout [12] could also be considered, we leave it as future work.

ACK reception. Consider the case when node i receives a valid ACK(a_k) from node j before the T_{H,b_k} timeout. If the corresponding PKT was unicasted and j was not the next hop, then the ACK is ignored. Otherwise, both estimators $s^a_{H,j}$ and $s^f_{H,j}$ are increased, and the ACK is rebroadcasted.

If the PKT was broadcasted, the behavior depends on whether it is the first ACK that was received, or not. In the former case, both estimators $s^a_{H,j}$ and $s^f_{H,j}$ are increased, and the ACK is rebroadcasted. Otherwise, only $s^a_{H,j}$ is increased and the ACK is not rebroadcasted.

For both broadcasts and unicasts, only one ACK is accepted per neighbor and PKT id b_k, the subsequent ACKs are ignored.

Rationale behind the dual reliability estimators. Keeping track of the first ACKs with $s^f_{H,j}$ is a performance optimization. The primary routing metric in Castor is the packet delivery reliability, but $s^f_{H,j}$ gives preference to lower round-trip routes that are typically shorter and consume less bandwidth. A similar method has been used with success in ARAN [7] and SRP [2]. Whereas, using the second estimator, $s^a_{H,j}$, to keep track of all ACKs allows Castor to obtain more routing information with one broadcasted PKT, leading to faster convergence under attacks and when exploring routes.

E. Flood Rate-Limiting

To protect the system from Denial-of-Service attacks exploiting the PKT flooding (§VI) Castor uses a PKT broadcast rate-limiting mechanism. The mechanism takes advantage of the fact that messages are neighbor-to-neighbor authenticated. For each neighbor $j = 1 \dots k$ the current allowed broadcast rate r_j is kept. The rates are initially set to one per second. When the broadcast is successful (i.e., when the ACK is received) the allowed rate is multiplied $\sigma_{\text{succ}} = 2$, otherwise it is multiplied by $\sigma_{\text{fail}} = 0.5$. The rate values are constrained to the $[r_{min} = 0.1s, r_{max} = 100s]$ range. The r_j rate limit is enforced by maintaining a size 3 leaky bucket rate-limiter for each neighbor [19]. The rate-limiting mechanism affects only the PKT broadcasts, PKT unicasts and all other messages are unaffected. We have found that this simple approach provides a strong defense against the flooding-based DoS attacks. We confirm that experimentally in §VII-E.

VI. Security analysis

We first show that Castor is resilient to general attacks relevant to any routing protocol, and then do the same for Castor-specific attacks. For presentation clarity, the discussion in this section assumes a static network. In the evaluation section (§VII) we demonstrate that Castor's security properties also hold under mobility. See the technical report [19] for a more elaborate version of this section.

A. General Attacks

Packet dropping. An adversarial node drops all (*blackhole attack*) or some (*grayhole attack*) of the packets it is expected to forward. The fraction of packets a grayhole drops can vary over time and dropping can be selective, affecting only specific type of packets.

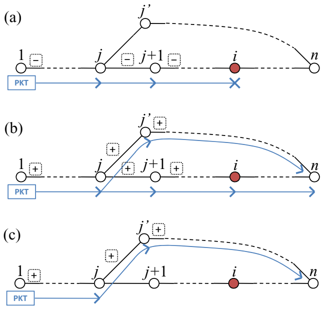

Fig. 1. **Dropping attack defence.** Reliability estimator increase indicated by "+", decrease by "-".

Consider a packet flow with id H from some correct source 1 to some correct destination n. Assume that the packets are forwarded by nodes $1, 2, \dots, n-1, n$, and that one of the nodes, say i, is adversarial (Fig. 1).

If i drops a PKT, n does not receive it and does not respond with an ACK. Our acknowledgment mechanism guarantees that n is the only node able to generate an ACK for the PKT. On PKT loss (Fig. 1(a)) every node $j = 1, \dots, i-1$ preceding i on the route times out waiting for the ACK and decreases its reliability estimator $s_{H,j+1}$ for the successor on the route. The more aggressively i drops, the lower the estimators become. One of the following eventually happens for some $j = 1, \dots, i-1$: j broadcasts the packet to all of its neighbors (Fig. 1(b)) or the reliability estimator $s_{H,j'}$ of some neighbor $j' \neq j+1$ of j exceeds $s_{H,j+1}$, and j forwards subsequent packets to j' (Fig. 1(c)). In Fig. 1(b), some neighbors of j succeed in delivering the PKT and responding with a correct ACK; j increases their reliability estimators and new routes are established. Eventually, after another packed drop, j re-routes to j', *away* from the source of unreliability. This is the fundamental mechanism through which Castor removes lossy nodes from the routes.

The behavior is similar if the adversarial node i forwards PKTs, but drops ACKs: Nodes $j = 1, \dots, i-1$ timeout waiting for the ACK and the same mechanism ensures that i is removed from the routes. The only difference to the PKT dropping is that the successors of i on the route receive the ACK and increase their respective reliability estimators. This is in fact not undesirable, as the resultant routing state update is correct.

Jamming. Adversarial nodes can prevent communication within their respective ranges: The attack can be mounted on the physical layer or the MAC layer, continuously or intermittently and selectively. Flows ending in the jammed regions are effectively denied communication. For other flows, a jammed region appears as a cluster of black- or gray-hole nodes and Castor is able to route around them.

Wormholes and tunnels. In a tunnel attack, remote adversarial nodes use their fast links to transfer messages out-of-band, appearing to be neighbors. In the more powerful wormhole attack, the out-of-band links are used to almost instantly relay, without any modification, messages received in one location to another remote location in the network. Thus, every node at one end of the wormhole believes itself to be a neighbor of every node at the other end. Route discovery mechanisms optimizing for hop-count or response time (as Castor does) are attracted by such "shortcuts" and wormholes and tunnels are likely to become part of many routes. The adversary can take advantage of this and take control over a large fraction of the traffic, which can then be maliciously dropped or corrupted.

Castor uses the tunnels and wormholes opportunistically as long as they allow the traffic to pass through. As soon as the attacker starts dropping the packets, the PKT-ACK loop is broken and Castor turns to alternative, more reliable neighbors for routing, avoiding the lossy tunnels and wormholes. We demonstrate this property experimentally in §VII-B.

Rushing attack. Even without fast out-of-band links, the adversarial nodes can attempt to place themselves on the routes. In a [20], nodes forward broadcasted PKTs as soon as possible by e.g., obtaining priority on the MAC layer, exploiting its vulnerabilities. From our perspective, this attack is a weaker variant of a wormhole/tunnel attack: When the rushing nodes start dropping, they will be routed around.

Sybil attack. In a Sybil attack, a single adversarial node appears as multiple nodes to its neighbors, using the cryptographic material of other compromised nodes. As reliability estimators are kept on a per-neighbor basis, routing around a Sybil node is harder: its neighbors have to decrease their reliability estimator for each of the identities of the Sybil node. The Sybil attack is not very different from the wormhole attack. In both attacks, the outcome is identical, a node gains a number of false neighbors. These can potentially become droppers; and when they do, they are detected and routed around. In the evaluation (§VII) we focus on the most severe attack in this class of neighborhood attacks: the wormhole attack.

B. Castor-Specific Attacks

Importance of flow isolation. Castor maintains reliability estimators per-flow, not per-destination, and it uses flow authenticators to identify the flows and cryptographically binds PKTs and ACKs. This ensures that 1) the in-network states for each of the flows are logically isolated from one another and 2) only the messages originating at the source or the destination can influence the flow's state. Without this, an attacker could generate false PKT-ACK pairs for any chosen flow and maliciously modify the routing state on all the nodes that the false PKTs and ACKs traverse. This could be used to, for example, prevent PKT delivery to a legitimate destination by re-routing the traffic to an adversary-controlled node.

Message corruption and forgery. The adversary can attempt to corrupt any field of ACKs or PKTs. If the payload M is modified, the data integrity verification fails at the destination and an ACK is not sent back to the source for that packet. Thus, any node corrupting the payload appears to its neighbors as a packet dropper, and it is routed around.

Replay attacks. As explained in §V-B, the adversary cannot successfully forge flow authenticators in PKTs or ACK authenticators. However they can be replayed. Several variants of replay attacks are possible. The objective is to influence state corresponding to legitimate active flows and attempt to reroute the PKTs in order to discard or corrupt them.

First, an adversary who does not modify the replay PKTs creates correct routing state (at worst redundant). This has no negative impact on the network performance. The adversary could, however, modify in every replayed PKT the parts that intermediate nodes cannot recognize as invalid: $E_{K_{sd}}(a_k)$ and M. The Castor nodes forward every distinct copy of the PKT even if the flow authenticators are identical. This ensures the correct PKT copy will get through even though nodes also receive the malformed clones.

Considering ACK replays, observe that a given ACK can be used to increase an estimator at node j for some neighbor k at most once. Hence, it is not possible to artificially increase an estimator by repeatedly replaying an ACK from one neighbor to another. Each correct node also ignores ACKs that correspond to PKTs it never forwarded. In addition, if a PKT was unicasted to some neighbor j, the corresponding ACK from any node other than j is ignored.

Finally, the adversary could, using its fast communication links, "reenact" a correct flow in some other part of the network. However, this attack cannot be sustained without delivering the PKTs to the correct destination in order to obtain the fresh valid ACKs needed for reenacting the flow, which defeats the purpose of the attack. See the technical report [19] for more details.

Flooding attack. In Castor, the nodes broadcast the PKTs whenever no reliable route is known. An adversary could use the PKT rebroadcasts as an attack amplification device. A high-rate stream of PKTs could be injected, each PKT belonging to a distinct new dummy flow. This would trigger a network-wide flood for each PKT, potentially causing global bandwidth starvation. All the routing protocols relying on flooding for route discovery have this vulnerability: an attacker can trigger floods at a high rate and cause Denial-of-Service. Very few of the existing protocols address this issue. Castor uses flood rate-limiting (§V-E) that limits how often a given neighbor can cause a PKT broadcast. We show experimentally (§VII-E) that this simple measure gives a high level of protection from the flooding attacks.

A neighbor is allowed a higher rate when the broadcasts that it causes are followed by ACKs. An attacker could exploit this and set up a colluding node in the network that would be ACKing the bogus PKT stream and thus allowing a higher attack rate. However, this also means that there would be enough bandwidth left for some of the benign messages, thus weakening the attack. In our experiments we found that this attack variant is much more difficult to perpetrate and it has a lower impact on performance than the simple non-ACKed PKT flooding. The rate limiters could be precisely tuned to provide a very strong defence against the ACKed flooding, but we leave it for future work.

VII. Performance Evaluation

Our evaluation is carried out using the ProtoPeer (`http://protopeer.net`) message passing framework, with JiST/SWANS (`http://jist.ece.cornell.edu/`) employed for MANET modeling. Apart from Castor, we have implemented three other routing protocols: Sprout, SRP and SEAD. We use the default-setting implementation of AODV

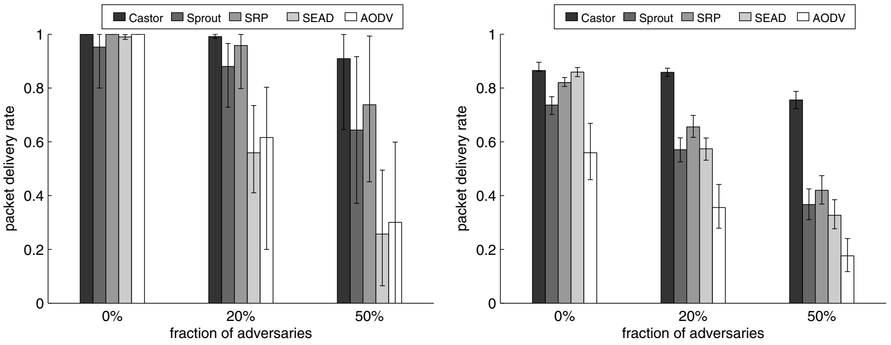

Fig. 2. **Blackhole attack resilience.** We vary the fraction of blackholes in the system without (left) and with (right) mobility.

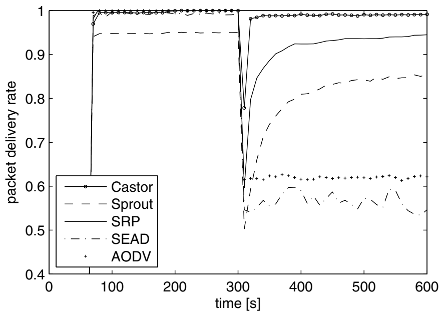

Fig. 3. **Blackhole attack resilience.** Failure recovery time.

from the JiST/SWANS library. The choice of protocols covers all combinations of on-demand/proactive and distance-vector/link-state categories. Our Sprout implementation uses the parameters recommended in [12], with one exception: to handle mobility, the maximum number of routes stored at a given time is set to 50. When exceeded, the lowest-ranked route is removed. The SSP [1] protocol is layered on top of SRP [2]. The experimental setup parameters are summarized in Table I.

We use the random waypoint mobility model (Table I). Nodes send neighbor discovery beacons every $250ms$. A node is removed from the neighbor set if not heard from for $1s$. Unless otherwise stated, the measurements are averaged over 50 independent runs, 1 hour of simulated time each. Every data point is a 10-second average. Each of the 50 runs has distinct node trajectories. The same trajectories are repeated for all the protocols. We show 90% confidence intervals.

TABLE I
EXPERIMENTAL SETUP

General			
plane size		3km by 3km	
nodes		100, placed uniformly at random, 10 radio neighbors on average	
MAC		802.11b at 1Mbps	
number of flows		5, source-destination disjoint	
flow rate		constant bit rate, 4 packets/s	
packet payload size		256 bytes	
Random waypoint mobility		**Castor**	
min. speed	1 m/s	γ	8
max. speed	20 m/s	δ	0.8
pause time	0s	T_{ACK}	500ms
SEAD			
periodic route update interval	5s		
SRP		**Sprout**	
α, β, δ	0.5	γ	1.25
r_s^{thr}	0	α_{pdr}	0.9
r_s^{max}	1	α_{rtt}	0.9

A. Dropping Attack

We implement selective blackholes, dropping all data packets, but allowing control packets (route discovery) through. For Castor, the attacker drops unicasted PKTs, and forwards broadcasted PKT to attract more routes. Fig. 2 shows the achieved PDR, as a function of the fraction of adversarial nodes. Recall that we look at the network-layer PDR, i.e., there are no retries to mask the packet loss; accordingly, for SSP we set $Retry_{MAX} = 0$.

The AODV and SEAD protocols do not make any end-to-end checks for packet loss and are unaware of the blackholes. The performance of the two protocols is thus significantly affected. Sprout and SRP monitor route reliability, and thus significantly improve over SEAD and AODV. However, Sprout and SRP do per-route performance accounting; under high fractions of attackers, most of the routes contain at least one adversarial node and both protocols take longer time to converge on a blackhole-free route. In contrast, Castor keeps reliability records with higher granularity, per-link instead of per-route, and can detect and route around the attackers much faster.

Failure recovery time. The benefits of storing reliability information per-link-per-flow, rather than per-route are clearly demonstrated in the following experiment (Fig. 3): 20% of the nodes become blackholes 5 minutes into the simulation. Castor recovers in under 30s, much faster than SRP or Sprout. Even after 30 minutes, Sprout and SRP fail to find reliable routes for some of the flows, which is reflected in the 50-run averages. AODV and SEAD, not surprisingly, do not recover from the attack.

Mobility. Although mobility degrades the performance of all protocols, Castor is the least affected: Nodes observe the topology changes locally and most of the time they are able to select an alternative next hop on the spot or perform a local flood to repair the route. In contrast, the other protocols must either: (i) wait for the new topology information to propagate through the network (Sprout and SEAD) or (ii) wait for the on-demand route (re)discovery to finish (AODV, SRP). In addition, the newly discovered routes must be "evaluated" (Sprout, SRP) for the presence of the black holes.

Bandwidth utilization. Bandwidth utilization for the performed experiments is shown in Fig. 4. For a fair comparison, we consider both the data and the control traffic. Despite the fact that Castor includes the full 256 byte data payload in the flooded PKTs, it often responds to changing network conditions by simple re-routing or limited flooding (§V-C), which results in amortized bandwidth cost comparable to the other protocols.

The two proactive protocols, Sprout and SEAD, require additional bandwidth for propagating the network topology information. Under mobility, even in the benign case, Sprout uses a substantial amount of bandwidth for link-state updates.

Delay. We have also measured the packet delivery delay

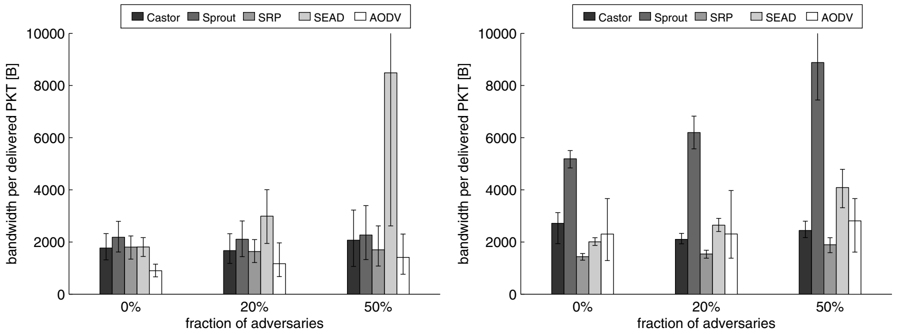

Fig. 4. **Bandwidth utilization under the blackhole attack.** Left: no mobility. Right: mobility. The experimental setup identical to Fig. 2.

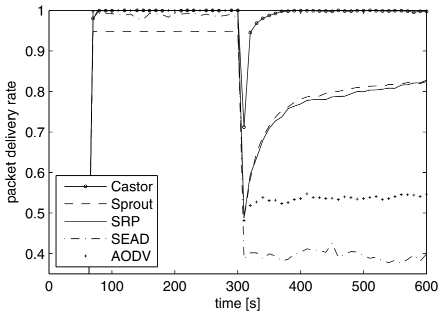

Fig. 5. **Wormhole attack resilience.** Failure recovery time.

and observed no significant differences between Castor and the other protocols. For details, see [19].

Grayhole attacks. We have also evaluated all the protocols under various variants of grayhole attacks, as well as pure blackholes dropping all packets. The overall behavior across all protocols was similar, with the average PDR higher but recovery time longer. We omit the details here, the complete set of results is available in the technical report [19].

B. Wormhole Attack

We set up a wormhole with three exit points forming an equilateral triangle, each pair of points separated by 1000m. The wormhole is implemented at the radio layer. The wormhole exit transceivers are the same as in the other nodes. Initially, the wormhole forwards all the packets. At the 5 minute mark, the wormhole stops retransmitting any data traffic, but still keeps retransmitting the control traffic and broadcasted PKTs. Out of all the evaluated wormhole behaviors, this one had the most severe impact on the performance of all the protocols. We measure how the PDR changes in response to the attack (Fig. 5).

All protocols except Castor fail to recover completely from the wormhole attack. The reasons are similar as for the blackhole attack: ADOV and SEAD do not recover as they continue to route through the lossy wormhole, whereas SRP and Sprout route around the wormhole but they are slower and less successful in finding adversary-free routes than Castor.

Note that Sprout has not been designed to defend against wormhole attacks [12]; instead, its authors recommend relying on solutions such as TrueLink [21]. We did not simulate TrueLink. Castor recovers from wormholes without any additional wormhole defense mechanisms and their overhead.

Other attacks. We did not evaluate tunnel, rushing or Sybil attacks, as from our perspective they are very similar in nature, though weaker, than the wormhole attack. We also omit replay attacks; as argued in §VI-B, they are not a significant threat.

C. Performance under Mobility

To test how node mobility influences the protocols' ability to detect the adversary, we set the number of blackholes to 20%, and vary the node pause time in the random waypoint model. We measure the PDR (Fig. 6).

Castor's local failure detection and repair can rapidly reroute the PKTs, when nodes go beyond the radio range or the new neighbor turns out to be a black hole. Sprout and SRP need to route more PKTs to determine which routes are reliable and often this process is slower than the rate of change in the topology. AODV and SEAD display constant performance, confirming that it is not the mobility, rather the attack, that affects them.

D. Scalability

We next measure the performance of the protocols for different network sizes, keeping node density constant. 20% of the nodes are blackholes under zero pause time mobility. The results are shown in Fig. 7.

The routing paths become longer with the increasing network size, thus finding an adversary-free path becomes more challenging. Longer paths are also more likely to break due to mobility. Under these conditions Castor still outperforms the other protocols and maintains a 60% packet delivery rate on a 6km by 6km plane with 400 mobile nodes and 80 blackholes. With increased size and mobility, Castor resorts to PKT flooding more often, which the bandwidth measurements confirm. The bandwidth utilization of the proactive protocols (SEAD and Sprout) significantly increases. At 400 nodes Sprout experiences a congestion collapse as the network is over-flooded with link-state updates.

E. Flooding Attack Resilience

Without the flood rate-limiting (FRL) mechanism (§V-E) Castor is vulnerable to the flooding attack (§VI-B). In what follows, we experimentally demonstrate the ability of FRL to thwart such an attack.

An attacker controls a single node, which starts 200 new dummy flows per second by broadcasting PKTs, each PKT containing a new unique flow authenticator. With FRL inactive, the attack prevents a large fraction of the traffic from passing through (Fig. 8). With FRL active, however, the nodes quickly detect and contain the attacker and the attack's impact is greatly reduced. Complete recovery may not be possible; in some cases the source or destination reside in the neighborhood of the rapidly broadcasting attacker, which effectively prevents local communication. In a larger network, the routing paths are longer on average and it is easier for the malicious PKT flood to disrupt them. However, when FRL is active in the larger network, the attack is contained to a smaller area relative to the whole plane size and the performance decrease is smaller.

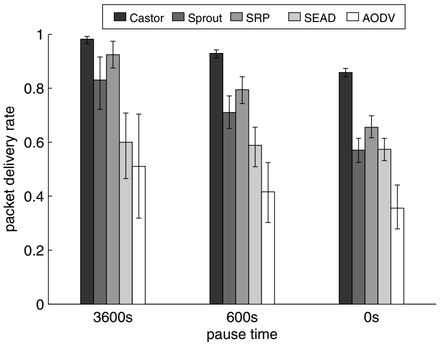

Fig. 6. **Performance under mobility.** 20% of the nodes are blackholes.

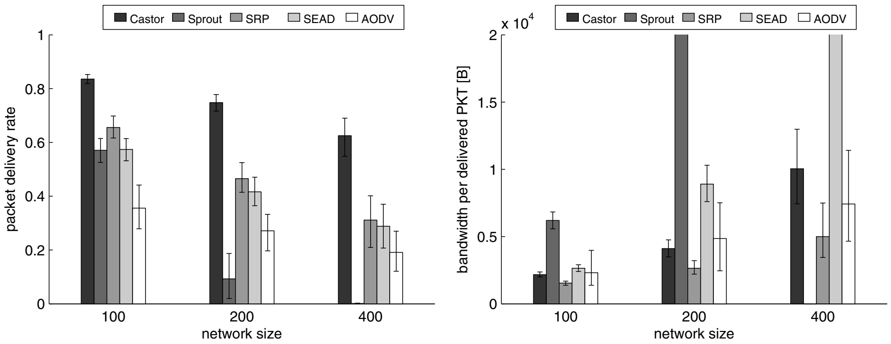

Fig. 7. **Scalability.** 20% of the nodes are blackholes.

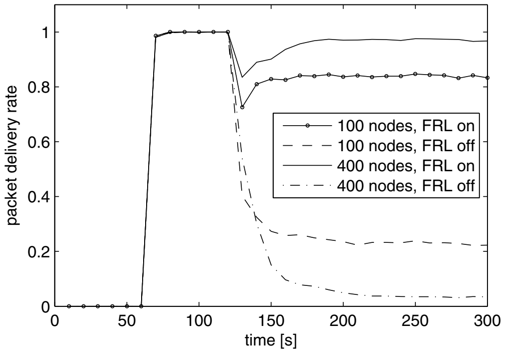

Fig. 8. **Flooding attack resilience.**The attack begins at the 120s mark. The five 4pkt/s flows begin at 60s. The curves are averages over 50 independent randomly seeded runs.

F. Congestion Control

Unlike other protocols, Castor includes the data payload in broadcast packets. But still it achieves the same or lower amortized bandwidth cost (§VII-A). Nonetheless, in our experiments, we observed bandwidth usage spikes during flow (re)establishment, which in turn may cause excessive ACK loss and lead to flow failures. This is especially so with high-rate flows that bring the network closer to saturation. The solution would be congestion control mechanisms that rate-limit sources, notably during flow (re)establishment. We leave this as future work.

VIII. Conclusions

We have proposed Castor, a novel secure communication protocol for ad hoc networks. Despite the very simple PKT-ACK messaging, the protocol is more resilient to attacks than any previously proposed secure communication protocols, especially in large and mobile networks, as demonstrated by the extensive comparative evaluation. All that is achieved relies on weak, and thus more practical, trust assumptions.

Several interesting open issues remain, among them: How would Castor interact with a reliable transfer protocol? Can the PKTs and ACKs be taken advantage of for cross-layer design? How can we tune the parameters of Castor, notably the ones controlling the broadcast vs. unicast behavior, to achieve the right balance between route exploration vs. exploitation? Could we extend the reliability estimators to measure both loss and delay? Finally, how do the potential solutions to the above problems affect the protocol security?

References

[1] P. Papadimitratos and Z. J. Haas, "Secure Data Communication in Mobile Ad Hoc Networks," *IEEE JSAC*, vol. 24, no. 2, 2006.

[2] P. Papadimitratos and Z. Haas, "Secure routing for mobile ad hoc networks," in *SCS CNDS'02*.

[3] Y. Hu, A. Perrig, and D. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," *Wireless Networks*, vol. 11, no. 1, 2005.

[4] G. Acs, L. Buttyan, and I. Vajda, "Provably secure on-demand source routing in mobile ad hoc networks," *IEEE TMC*, vol. 5, no. 11, pp. 1533–1546, 2006.

[5] P. Papadimitratos and Z. Haas, "Secure Link State Routing for Mobile Ad Hoc Networks," in *Proceedings of the IEEE Workshop on Security and Assurance in Ad Hoc Networks*, 2003.

[6] P. Papadimitratos, Z. Haas, and J. Hubaux, "How to Specify and How to Prove Correctness of Secure Routing Protocols for MANET," in *IEEE BROADNETS'06*.

[7] K. Sanzgiri, B. Dahill, B. Levine, C. Shields, and E. Belding-Royer, "A secure routing protocol for ad hoc networks," in *IEEE ICNP'02*.

[8] M. Zapata, "Secure ad hoc on-demand distance vector routing," *ACM Mobile Computing and Communications Review*, vol. 6, no. 3, 2002.

[9] C. Perkins, E. Belding-Royer, S. Das *et al.*, "Ad hoc on-demand distance vector (AODV) routing," RFC 3561, 2003.

[10] Y. Hu, D. Johnson, and A. Perrig, "SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks," *Ad Hoc Networks*, vol. 1, no. 1, pp. 175–192, 2003.

[11] P. Papadimitratos and Z. Haas, "Secure message transmission in mobile ad hoc networks," *Ad Hoc Networks*, vol. 1, no. 1, pp. 193–209, 2003.

[12] J. Eriksson, M. Faloutsos, and S. Krishnamurthy, "Routing amid colluding attackers," in *IEEE ICNP'07*, 2007.

[13] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H. Rubens, "Odsbr: An on-demand secure byzantine resilient routing protocol for wireless ad hoc networks," *ACM TISSEC*, vol. 10, no. 4, 2008.

[14] L. Buttayan and J.-P. Hubaux, *Security and Cooperation in Wireless Networks*. Cambridge University Press, 2007.

[15] G. Di Caro and M. Dorigo, "AntNet: Distributed stigmergetic control for communications networks," *Journal of AI Research*, vol. 9, no. 2, pp. 317–365, 1998.

[16] S. Marwaha, C. Tham, and D. Srinivasan, "A novel routing protocol using mobile agents and reactive route discovery for ad hoc wireless networks," in *ICON'02*.

[17] O. Hussein and T. Saadawi, "Ant routing algorithm for mobile ad-hoc networks (ARAMA)," in *IEEE IPCCC'03*, pp. 281–290.

[18] A. Perrig, R. Canetti, J. Tygar, and D. Song, "The TESLA broadcast authentication protocol," *RSA CryptoBytes*, vol. 5, no. 2, pp. 2–13, 2002.

[19] W. Galuba, P. Papadimitratos, M. Poturalski, K. Aberer, Z. Despotovic, and W. Kellerer, "More on castor: the scalable secure routing protocol for ad-hoc networks," EPFL, Tech. Rep. LSIR-REPORT-2009-002, 2009.

[20] Y. Hu, A. Perrig, and D. Johnson, "Rushing attacks and defense in wireless ad hoc network routing protocols," in *ACM WiSe'03*.

[21] J. Eriksson, S. Krishnamurthy, and M. Faloutsos, "Truelink: A practical countermeasure to the wormhole attack in wireless networks," in *IEEE ICNP'06*.