Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols

Yih-Chun Hu
Carnegie Mellon University
yihchun@cs.cmu.edu

Adrian Perrig
Carnegie Mellon University
perrig@cmu.edu

David B. Johnson
Rice University
dbj@cs.rice.edu

ABSTRACT

In an *ad hoc network*, mobile computers (or nodes) cooperate to forward packets for each other, allowing nodes to communicate beyond their direct wireless transmission range. Many proposed routing protocols for ad hoc networks operate in an *on-demand* fashion, as on-demand routing protocols have been shown to often have lower overhead and faster reaction time than other types of routing based on periodic (proactive) mechanisms. Significant attention recently has been devoted to developing secure routing protocols for ad hoc networks, including a number of secure on-demand routing protocols, that defend against a variety of possible attacks on network routing. In this paper, we present the *rushing attack*, a new attack that results in denial-of-service when used against *all* previous on-demand ad hoc network routing protocols. For example, DSR, AODV, and secure protocols based on them, such as Ariadne, ARAN, and SAODV, are unable to discover routes longer than two hops when subject to this attack. This attack is also particularly damaging because it can be performed by a relatively weak attacker. We analyze why previous protocols fail under this attack. We then develop *Rushing Attack Prevention (RAP)*, a generic defense against the rushing attack for on-demand protocols. RAP incurs *no cost* unless the underlying protocol fails to find a working route, and it provides provable security properties even against the strongest rushing attackers.

Categories and Subject Descriptors: C.0 [**Computer-Communications Networks**]: Security and protection; C.2.2 [**Network Protocols**]: Routing Protocols

General Terms: Security, Performance

Keywords: Ad hoc network routing, security, routing, rushing

This work was supported in part by NASA under grant NAG3-2534, by NSF under grant FD99-79852, by DARPA under contract N66001-99-2-8913, by the Center for Computer and Communications Security at Carnegie Mellon under grant DAAD19-02-1-0389 from the Army Research Office, and by a gift from Bosch and Schlumberger. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of NASA, USPS, NSF, DARPA, ARO, Bosch, Schlumberger, Carnegie Mellon University, Rice University, or the U.S. Government or any of its agencies.

WiSe 2003, September 19, 2003, San Diego, California, USA.

1. INTRODUCTION

An *ad hoc network* is a collection of mobile computers (or nodes) that cooperate to forward packets for each other to extend the limited transmission range of each node's wireless network interface. A routing protocol in such a network finds routes between nodes, allowing a packet to be forwarded through other network nodes towards its destination. In contrast to traditional network routing protocols, for example for wired networks, ad hoc network routing protocols must adapt more quickly, since factors such as significant node movement and changing wireless conditions may result in rapid topology change.

This problem of routing in ad hoc networks is an important one, and has been extensively studied. This study has resulted in several mature protocols [10, 21, 31, 33]. Ad hoc networks are targeted at environments where communicating nodes are mobile, or where wired network deployment is not present or not economical. Many of these applications may run in untrusted environments and may therefore require the use of a secure routing protocol. Furthermore, even when the presence of an attacker is not forseen, a secure ad hoc network routing protocol can also provide resilience against misconfigured nodes. In the current Internet, for example, misconfigured routing tables contribute to the majority of routing instabilities [27]. Similarly, a software or hardware failure should cause only the affected node to fail, and not perturb the stability of routing in the remainder of the network. Mission or safety-critical networks can use secure ad hoc routing protocols so that configuration errors, software bugs, or hardware failures do not disturb routing at other nodes. As a result, several secure ad hoc network routing protocols have been proposed [7, 14, 17, 32, 37, 40, 46].

In this paper, we present a new attack, the *rushing attack*, which results in denial-of-service when used against all previously published on-demand ad hoc network routing protocols. Specifically, the rushing attack prevents previously published secure on-demand routing protocols to find routes longer than two-hops (one intermediate node between the initiator and target).

Because on-demand protocols generally have lower overhead and faster reaction time than other types of routing based on periodic (proactive) mechanisms, on-demand protocols are better suited for most applications. To defend this important class of protocols against the rushing attack, we develop a generic secure Route Discovery component, called *Rushing Attack Prevention (RAP)*, that can be applied to any existing on-demand routing protocol to allow that protocol to resist the rushing attack.

Our main contributions in this paper are the presentation of the rushing attack, the development and analysis of our new secure Route Discovery component that demonstrates that it is possible to secure against the rushing attack, and a general design that uses this component to secure any on-demand Route Discovery mecha-

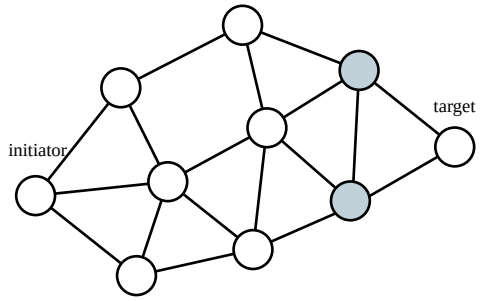


Figure 1: Example network illustrating the rushing attack.

nism against the rushing attack.

In Section 2 of this paper, we introduce the rushing attack. Section 3 details our assumptions. Section 4 describes our Secure Neighbor Detection and Secure Route Discovery procedures, and Section 5 presents two evaluations of our Route Discovery component: a simulation study of the performance of our mechanisms, and an analytical evaluation that gives a conservative lower bound on the probability that our protocols discover a working route when subject to this attack. In Section 6, we discuss related work, and in Section 7, we present conclusions.

2. THE RUSHING ATTACK AGAINST AD HOC NETWORK ROUTING PROTOCOLS

We introduce here a new attack, which we call *the rushing attack*, that acts as an effective denial-of-service attack against all currently proposed on-demand ad hoc network routing protocols, including protocols that were designed to be secure. In an on-demand protocol, a node needing a route to a destination floods the network with ROUTE REQUEST packets in an attempt to find a route to the destination. To limit the overhead of this flood, each node typically forwards only one ROUTE REQUEST originating from any Route Discovery. In particular, existing on-demand routing protocols, such as AODV [33], DSR [21], LAR [24], Ariadne [17], SAODV [46], ARAN [40], AODV secured with SUCV [7], and SRP [32], only forward the REQUEST that arrives *first* from each Route Discovery. In the rushing attack, the attacker exploits this property of the operation of Route Discovery.

We now describe the rushing attack in terms of its effect on the operation of DSR Route Discovery [19, 20, 21]; other protocols such as AODV [34], Ariadne [17], SAODV [46], and ARAN [40] are vulnerable in the same way. In the network shown in Figure 1, the initiator node initiates a Route Discovery for the target node. If the ROUTE REQUESTs for this Discovery forwarded by the attacker are the first to reach each neighbor of the target (shown in gray in the figure), then any route discovered by this Route Discovery will include a hop through the attacker. That is, when a neighbor of the target receives the rushed REQUEST from the attacker, it forwards that REQUEST, and will not forward any further REQUESTs from this Route Discovery. When non-attacking REQUESTs arrive later at these nodes, they will discard those legitimate REQUESTs. As a result, the initiator will be unable to discover any usable routes (i.e., routes that do not include the attacker) containing at least two hops (three nodes).

In general terms, an attacker that can forward ROUTE REQUESTs more quickly than legitimate nodes can do so, can increase the probability that routes that include the attacker will be discovered rather than other valid routes. Whereas the discussion above has used the case of nodes that forward only the *first* ROUTE REQUEST from any Route Discovery, the rushing attack can also be used against any protocol that predictably forwards *any* particular REQUEST for each Route Discovery.

A rushing attacker need not have access to vast resources. On-demand routing protocols delay ROUTE REQUEST forwarding in two ways. First, Medium Access Control (MAC) protocols generally impose delays between when the packet is handed to the network interface for transmission and when the packet is actually transmitted. In a MAC using time division, for example, a node must wait until its time slot to transmit, whereas in a MAC using carrier-sense multiple access, a node generally performs some type of backoff to avoid collisions; protocols like IEEE 802.11 also impose an interframe spacing time before transmission actually begins. Second, even if the MAC layer does not specify a delay, on-demand protocols generally specify a delay between receiving a REQUEST and forwarding it, in order to avoid collisions of the REQUEST packets. In particular, because REQUEST packets are broadcast, and collision detection for broadcast packets is difficult, routing protocols often impose a randomized delay in REQUEST forwarding. An attacker ignoring delays at either the MAC or routing layers will generally be preferred to similarly situated non-attacking nodes. One way to thwart an attacker that rushes in this way is to remove these delays at both the MAC and routing layers, but this approach does not work against all types of rushing attackers and is not general. For example, in a dense network using a CSMA MAC layer, if a node *A* initiates a Route Discovery, and *B* is two hops away from *A*, and *C* and *D* are neighbors of both *A* and *B*, then then *B* will likely not receive the ROUTE REQUEST due to a collision between REQUESTs forwarded by *C* and *D*. In a dense network, such collisions may often prevent the discovery of any nontrivial routes (routes longer than a direct link), which is even more severe than the rushing attack, which prevents the discovery of routes longer than two hops.

Another way that a relatively weak attacker can obtain an advantage in forwarding speed is to keep the network interface transmission queues of nearby nodes full. For example, if each node processes the packets it receives in order, and an inefficient REQUEST authentication mechanism is used, the attacker can keep other nodes busy authenticating REQUESTs containing bogus authentication, thus slowing their ability to forward legitimate REQUESTs. Protocols employing public key techniques are particularly susceptible to these attacks, since they require substantial computation to validate each received REQUEST.

A relatively weak attacker can also achieve faster transit of its REQUEST packets by transmitting them at a higher wireless transmission power level, thus reducing the number of nodes that must forward that REQUEST to arrive at the target. Since packet transit time at each hop is dominated by the processing time at the forwarding node, reducing the path to the target by just one hop is likely to provide a significant latency advantage, thus strengthening the attackers position.

A more powerful rushing attacker may employ a wormhole [15] to rush packets. In this case, the attacker simply forwards all control packets (but not data packets) received at one node (the attacker) to another node in the network (e.g., a second attacker). This forms a tunnel in the network, where packets reaching one end of the tunnel are broadcast out the other end. If the tunnel provides significantly faster transit than legitimate forwarders, nodes near one end of the tunnel generally will be unable to discover working routes to the other end of the tunnel, since it will generally discover routes through the tunnel. In general, a wired tunnel (in which the two attackers have a wired connection between themselves) will provide faster transit than native wireless (multihop) forwarding, since

node processing delay in forwarding is much longer than the propagation time.

The rushing attack applies to all proposed on-demand protocols because such protocols must limit the number of packets that any node will transmit in response to a single Route Discovery. Currently proposed protocols choose to forward at most one REQUEST for each Discovery; any protocol that allows an attacker to predict which ROUTE REQUEST(s) will be chosen for forwarding at each hop will be vulnerable to some variant of the rushing attack.

3. ASSUMPTIONS

3.1. Network Assumptions

We make the common assumption that most network links are bidirectional. More specifically, we require that the network remain connected when unidirectional links are ignored. Our Secure Neighbor Detection protocol rejects unidirectional links, so underlying routing protocols can assume that the network is free of unidirectional links. If another Secure Neighbor Detection technique is used, and that technique supports unidirectional links, then the ability of our Secure Route Discovery mechanism to discover and use unidirectional links is limited only by the underlying routing protocol.

Wireless physical layers for sending data from one node to another are often vulnerable to jamming. Mechanisms such as spread spectrum modulation [38], or directional antennas have been extensively studied as means of improving resistance to physical jamming. In addition, an effective jamming attack usually requires additional hardware; in contrast, a rushing attack is much simpler to do because the attacker can use the same hardware as legitimate nodes. An attacker can even remotely break into a legitimate node and perform these attacks. Moreover, the rushing attack allows for far more selective denial-of-service, and is thus harder to detect. Jamming attacks are relatively broad (they deny service to a large number of participants) and are thus also easier to detect. Though a jamming attack is also an important denial-of-service attack, we present mechanisms to defend against the rushing attack because we believe that the rushing attack is more easily performed.

Medium Access Control protocols are also often vulnerable to attack. For example, in IEEE 802.11, an attacker can paralyze nodes in its neighborhood by sending Clear-To-Send (CTS) frames periodically, setting the "Duration" field of each frame equal to the interval between such frames [17]. Less sophisticated Medium Access Control protocols, such as ALOHA and Slotted ALOHA [1], are not vulnerable to such attacks but have lower efficiency. In this paper, we disregard attacks on Medium Access Control protocols.

Prior work has shown that ad hoc network routing in general does not scale well [11]. Most existing simulation of ad hoc network routing protocols consider scenarios of 50 to 500 nodes. In this work, we focus on such medium-sized networks, and will not consider scalability issues; however, we believe that mechanisms such as clustering, which improve the scalability of other on-demand ad hoc network routing protocols, can also improve the scalability of our approach.

3.2. Security Assumptions and Key Setup

The protocols discussed in this paper require an instantly-verifiable broadcast authentication protocol, for which we use a digital signature. However, any signature used should be able to keep up with verification at line speed, to avoid a denial-of-service attack where an attacker overwhelms the victim by flooding it with bogus messages. One example of a protocol which should be fast enough on many nodes is the the HORS one-time signature by Reyzin and Reyzin [39]. We use the constructions of the BiBa [36] one-time signature in conjunction with the HORS one-time signature to design an efficient instantly-verifiable broadcast authentication protocol. We also use a Merkle hash tree [29] to generate one signature over multiple messages, such that each message is independently verifiable. As used in our simulation evaluation, HORS requires an average of $156,760$ hashes per second to sign and verify all messages in a 100 node network, a rate easily achievable even by PDAs. We assume that the keys necessary for broadcast authentication are distributed in advance; a number of techniques for distributing such information have been proposed [2, 17, 18, 25, 43, 47]. To escape the circular dependency of secure routing and key distribution, Hu et al propose a simple routing protocol that discovers a route to a trusted third party, which can in turn bootstrap the initial keys [17].

If a wormhole attack, in which an attacker selectively tunnels packets from one place in the network to another, is considered a possible threat, our Secure Neighbor Detection requires a mechanism to detect such a tunnel between any two legitimate nodes. A number of mechanisms for preventing the wormhole attack, such as TIK, geographical leashes and RF watermarking, have been proposed. Depending on the mechanism used to implement packet leashes, this requirement benefit other parts of the protocol: TIK [15], for example, authenticates each packet in a lightweight manner, thus protecting the more expensive signature verification from a denial-of-service attack. In particular, if a node *A* receives an authenticated packet containing a bogus signature from node *B*, then *A* can lower the priority with which it checks signatures sent by *B*. As a result, an attacker can only cause each node to verify one bogus signature for each node compromised by that attacker.

We do not assume tamper-proof hardware; the attacker can thus compromise nodes and steal their cryptographic keys. We assume a powerful attacker, which we call *coordinated attacker*. This is an attacker that compromised multiple nodes (and thus knows all their cryptographic keys), with a fast channel to route packets amongst themselves.

4. SECURE ROUTING REQUIREMENTS AND PROTOCOL

In this section, we describe a set of generic mechanisms that together defend against the rushing attack: *secure Neighbor Detection*, *secure route delegation*, and *randomized* ROUTE REQUEST *forwarding*. We also describe a technique to secure any protocol using an on-demand Route Discovery protocol.

In previous on-demand protocols, node *B* considers node *A* to be a neighbor when *B* receives a broadcast message from *A*. Secure Neighbor Detection, which replaces standard Neighbor Detection, allows each neighbor to verify that the other is within a given maximum transmission range. Once a node *A* forwarding a ROUTE REQUEST determines that node *B* is a neighbor (that is, is within the allowable range), it signs a *Route Delegation* message, allowing node *B* to forward the ROUTE REQUEST. When node *B* determines that node *A* is within the allowable range, it signs an *Accept Delegation* message.

Randomized selection of the ROUTE REQUEST message to forward, which replaces traditional duplicate suppression in on-demand route discovery, ensures that paths that forward REQUESTs with low latency are only slightly more likely to be selected than other paths.

Figure 2 shows the basic design of our complete rushing attack prevention mechanism.

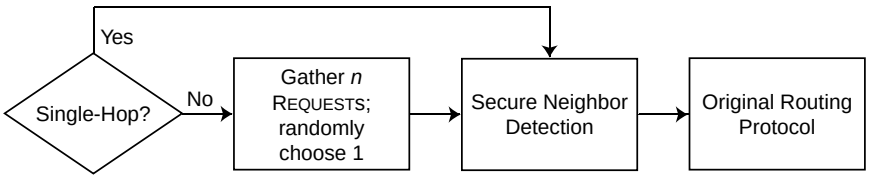


Figure 2: Our combined mechanisms to secure an on-demand route discovery protocol against the rushing attack.

4.1. Notation

We use the following notation:

- A or B denote communicating nodes.
- $A : \eta \xleftarrow{R} \{0,1\}^{\ell}$ denotes that node A randomly selects an ℓ-bit long nonce η.
- $A \rightarrow B : \langle M, H(A \,||\, \eta)\rangle$ means that node A sends B the message M and the hash of A's identifier concatenated with the nonce η.
- $A \rightarrow * : \langle M, \Sigma_M\rangle$ means that node A broadcasts message M with its signature Σ_M.

4.2. Secure Neighbor Detection

One simple instance of the rushing attack is when an attacker forwards a ROUTE REQUEST beyond the normal radio transmission range (for example by using a higher gain antenna or a higher power level), thus suppressing subsequent REQUESTs from this Route Discovery. In this section, we present a secure Neighbor Detection protocol that allows both the sender and the receiver of a ROUTE REQUEST to verify that the other party is within the normal direct wireless communication range.

The functionality of *Neighbor Detection*, in which two nodes detect a bidirectional link between themselves, is present in some form in almost every routing protocol. For example, a node participating in a periodic protocol generally broadcasts advertisements, allowing its neighbors to detect it. Most on-demand routing protocols, on the other hand, perform Neighbor Detection implicitly. In those protocols, a node receiving a ROUTE REQUEST considers itself to be a neighbor of the previous-hop node that transmitted the REQUEST. When that node propagates the REQUEST, it claims a link between the transmitter and the recipient. Unfortunately, this implicit Neighbor Detection does not prevent an attacker node receiving a REQUEST from simply replaying it. In addition, if the address of the previous-hop node is unauthenticated, an attacker can claim to be any node propagating a REQUEST, and the next hop will trust that information (we call this the *repeater attack*). This repeater attack is serious, because two nodes that are not within communication range believe that the other is is its neighbor, giving the attacker the ability to selectively forward packets between the two nodes. The repeater attack is an instance of a *wormhole attack* [16].

Requirements for Secure Neighbor Detection. Two nodes detect each other as neighbors only if they can communicate and they are within some maximum transmission range. The secure Neighbor Detection protocol thus prevents an attacker from: (1) introducing two nodes that are not within the maximum transmission range as neighbors; and (2) claiming that it is a neighbor of another node without being able to hear packets directly from that node. From the first requirement, it follows that an attacker should not be able to *tunnel* a neighbor solicitation from one compromised node to another uncompromised node. The second requirement demands that a node (or an accomplice of that node) needs to hear the neighbor solicitation, since otherwise it cannot claim to be a neighbor. Finally, the protocol should not introduce a denial-of-service opportunity; for example, flooding a node with neighbor requests should not consume all CPU resources of that node.

Our Secure Neighbor Detection Protocol. We present a secure Neighbor Detection protocol that allows both the initiator and the responder to check that the other is within a maximum communication range. This protocol is similar to that proposed by Brands and Chaum for bounding the maximum distance between two nodes [6]. Assuming negligible MAC protocol delays, we design a simple three-round mutual authentication protocol that uses tight delay timing to ensure that the other party is within communication range. In the first round, the initiating node sends a Neighbor Solicitation packet, either by unicasting that packet to a specific neighbor, or by broadcasting the packet. Next, a node receiving the Neighbor Solicitation packet sends a Neighbor Reply packet. Finally, the initiator sends a Neighbor Verification, which includes broadcast authentication of a timestamp and the link from the source to the destination. Figure 3 shows an example of the protocol. If a node wishes to detect multiple neighbors, it must request a response from each neighbor, and must initiate Neighbor Detection with each neighbor separately, in order to avoid an implosion of Neighbor Reply packets.

To ensure freshness of the reply messages, we use nonces η_1 and η_2. The initiator picks η_1 at random (of sufficient length that an attacker has a negligible probability of guessing it) and is thus certain that the reply message is fresh if the received nonce matches η_1. The measured delay between sending the first message and receiving the second message provides an upper bound on the distance of the neighbor: given delay Δ, the neighbor node is no farther away than $\Delta/2 \times c$, where c is the speed of light. This is accurate if a node can quickly process the first message and return an authenticated second message; for example, if HORS is used for authentication, a node need only perform one hash function to authenticate the reply. The authentication on message M_2 ensures to the initiator that the response indeed comes from the correct responder. In the general case, we use the same digital signature for authentication, but if the two nodes share a secret key, we can also use a message authentication code for this purpose, for example HMAC [4]. Similarly, the nonce η_2 and the signature on message M_3 ensure to the responder that the initiator is within transmission range if message M_3 arrives after a sufficiently short delay.

Finally, we rate-limit New Neighbor Solicitations to prevent an attacker from flooding its neighbors. Figure 3 shows the full protocol.

Integration with an On-Demand Protocol. In an on-demand protocol, neighbor verification is performed during each Route Discovery. As a result, we can defend against New Neighbor Solicitation floods, by relying on the underlying protocol to defend against ROUTE REQUEST floods; a node responds to any New Neighbor

S: $\eta_1 \xleftarrow{R} \{0,1\}^\ell$
$M_1 = \langle \text{NEIGHBOR SOLICITATION}, S, \eta_1 \rangle$
$\Sigma_{M_1} = \text{Sign}(H(M_1))$
$S \rightarrow *$: $\langle M_1, \Sigma_{M_1} \rangle$

R: $\eta_2 \xleftarrow{R} \{0,1\}^{(\ell)}$
$M_2 = \langle \text{NEIGHBOR REPLY}, S, R, \eta_1, \eta_2 \rangle$
$\Sigma_{M_2} = \text{Sign}(H(M_2))$
$R \rightarrow S$: $\langle M_2, \Sigma_{M_2} \rangle$

S: $M_3 = \langle \text{NEIGHBOR VERIFICATION}, S, R, \eta_1, \eta_2 \rangle$
$\Sigma_{M_3} = \text{Sign}(H(M_3))$
$S \rightarrow R$: $\langle M_3, \Sigma_{M_3} \rangle$

Figure 3: Neighbor Detection between initiator S and responder R.

Solicitation presented with a valid REQUEST. If desired, REQUEST flood prevention can be achieved through the use of a hash chain, as in Ariadne [17]. In particular, in Ariadne, each node maintains a hash chain, and uses elements of the hash chain to authenticate the flooded REQUEST. These hash chain values provide cheap authentication, and a victim receiving too many Route Discoveries from an attacker can rate-limit forwarding of that attacker's REQUESTs. In RAP, we can instead use HORS or any other efficient authentication mechanism with this rate limiting, to prevent excessive flooding.

When a node A forwards a REQUEST, it includes in that REQUEST a broadcast Neighbor Solicitation. Each node B forwarding that REQUEST returns a Neighbor Reply, and piggybacks on the Neighbor Reply a unicast Neighbor Solicitation for A. If A decides that B is a neighbor based on the wormhole prevention mechanism used, A returns a signed Neighbor Verification that verifies the link from A to B. A also includes in packet a Neighbor Reply to the unicast Neighbor Solicitation sent by B. If B decides that A is a neighbor based on the wormhole prevention mechanism used, B forwards the REQUEST, including the Neighbor Verification for the $A \rightarrow B$ link signed by A, and also including a Neighbor Verification for the $B \rightarrow A$ link signed by itself. B need not return a Neighbor Verification, since A is likely to hear the forwarded REQUEST, which includes the $B \rightarrow A$ Neighbor Verification. Figure 4 shows how B forwards a REQUEST from A.

4.3. Secure Route Delegation

In our ROUTE REQUEST propagation, we want to enable each node to verify that all the secure Neighbor Detection steps were performed between any adjacent pair of nodes in the REQUEST, i.e., verify that *both* nodes of each adjacent node pair indeed believes to be a neighbor. We achieve this property through a *Secure Route Delegation* mechanism, which is inspired by the work of Kent et al. in S-BGP [22, 23]. S-BGP uses *Route Attestations* to ensure that each Autonomous System (AS) listed in the BGP AS path is indeed a valid AS. In S-BGP, before sending a route update to its neighbor, the AS signs a route attestation delegating it the right to further propagate the update.

We use this mechanism to enable the nodes to verify that all the secure neighbor detection protocols were executed and that *both* neighbors believe that they are within transmission range. We describe the protocol based on an example. Consider two neighboring nodes A and B, where A received the current ROUTE REQUEST originating from node S destined for node R with the sequence number *id*. Node A engages in the secure neighboring detection protocol and finds after the second message that B is indeed within range, so it delegates the ROUTE REQUEST to B as follows:

$M_A = \langle \text{ROUTE DELEGATION}, A, B, S, R, id \rangle$
$\Sigma_{M_A} = \text{Sign}(H(M_A))$
$A \rightarrow B$: $\langle \Sigma_{M_A} \rangle$

Node A does not need to send the message to B, as B can reconstruct all the fields of the message and verify the signature. The ROUTE DELEGATION message can be bundled together with the last message of the secure Neighbor Detection protocol. If B believes that A is indeed a neighbor within range, B will accept the ROUTE DELEGATION, continue the protocol, and sign another ROUTE DELEGATION with the next neighbor.

4.4. Randomized Message Forwarding

The secure Neighbor Detection and secure Route Delegation techniques are not sufficient to thwart the rushing attack, since an adversary can still get an advantage by forwarding ROUTE REQUESTs very rapidly. We use a random selection technique to minimize the chance that a rushing adversary can dominate all returned routes.

In traditional ROUTE REQUEST forwarding, the receiving node immediately forwards the REQUEST and suppresses all subsequent REQUESTs. In our modified flooding, a node first collects a number of REQUESTs, and selects a REQUEST at random to forward. There are thus two parameters to our randomized forwarding technique: first, the number of REQUEST packets to be collected, and second, the algorithm by which timeouts are chosen.

Given perfect information, each forwarding node would collect the maximum possible number of REQUESTs before forwarding one, since this approach provides the most effective defense against a rushing attack. However, when the number of REQUESTs is chosen to be too large, randomized forwarding will heavily rely on the timeout to trigger REQUEST forwarding, increasing latency and possibly reducing security. In a real network, perfect information is generally not available; as a result, initiators can include in each Route Discovery the number of REQUESTs to buffer before forwarding one, and can adjust this parameter adaptively, based on the REPLY latency and on the parameters chosen by other nodes. Alternatively, this number can be chosen as a global parameter, or locally using an adaptive algorithm, though an adaptive algorithm may allow certain new attacks.

When perfect topology information is available, the choice of timeout should be based on the number of legitimate hops between the initiator and the node forwarding the REQUEST; closer nodes should choose shorter timeouts than far-away nodes. This topological information can be approximated by location information; that is, nodes that are geographically closer should choose smaller timeouts than nodes that are geographically farther away. When geographic information is not available, nodes can randomly choose timeouts; however, this approach reduces security by favoring nodes choosing shorter timeouts.

4.5. Secure Route Discovery

In this section, we describe our secure route discovery protocol. We use three techniques in concert to prevent the rushing attack: our secure Neighbor Discovery protocol, our secure Route Delegation and delegation acceptance protocol, and randomized selection of which ROUTE REQUEST will be forwarded.

The intuition behind Secure Route Discovery is to make the forwarding of REQUEST packets less predictable by buffering the first n REQUESTs received, then randomly choosing one of those REQUESTs. However, we need to prevent an attacker from filling too many of these n REQUESTs, since otherwise the attacker could simply rush n copies of a REQUEST, rather than a single REQUEST, and our scheme would once again be vulnerable to the rushing attack.

A: $\eta_A \xleftarrow{R} \{0,1\}^\ell$
$M_{1a} = \langle \text{ROUTE REQUEST}, id, \ldots \rangle$
$M_{1b} = \langle \text{NEIGHBOR SOLICITATION}, A, \eta_A \rangle$
$\Sigma_{M_1} = \text{Sign}(H(M_{1a} \,||\, M_{1b}))$
$A \rightarrow *$: $\langle M_{1a}, M_{1b}, \Sigma_{M_1} \rangle$

B: $\eta_B \xleftarrow{R} \{0,1\}^\ell$
$M_{2a} = \langle \text{NEIGHBOR REPLY}, A, B, \eta_A, \eta_B \rangle$
$\Sigma_{M_2} = \text{Sign}(H(M_{2a}))$
$B \rightarrow A$: $\langle M_{2a}, M_{2b}, \Sigma_{M_2} \rangle$

A: $M_{3a} = \langle \text{NEIGHBOR VERIFICATION}, A, B, \eta_A, \eta_B \rangle$
$\Sigma_{M_{3a}} = \text{Sign}(H(M_{3a}))$
$M_{3b} = \langle \text{ROUTE DELEGATION}, A, B, S, R, id \rangle$
$\Sigma_{M_{3b}} = \text{Sign}(H(M_{3b}))$
$A \rightarrow B$: $\langle M_{3a}, \Sigma_{M_{3a}}, M_{3b}, \Sigma_{M_{3b}} \rangle$

B: $\eta'_B \xleftarrow{R} \{0,1\}^\ell$
$M_{4a} = \langle \text{ROUTE REQUEST}, id, \ldots, \Sigma_{M_{3b}}, \Sigma_{M_{4a}} \ldots \rangle$
$M_{4b} = \langle \text{NEIGHBOR SOLICITATION}, B, \eta'_B \rangle$
$\Sigma_{M_4} = \text{Sign}(H(M_{4a}) \,||\, H(M_{4b}))$
$B \rightarrow *$: $\langle M_{4a}, M_{4b}, \Sigma_{M_4} \rangle$

Figure 4: B forwarding the REQUEST from A. Σ_{M_2} can be generated using a shared key, if available. The ROUTE REQUEST in M_{4a} includes the bidirectional Neighbor Verification messages M_{3a} and M_{4c}, together with the necessary authenticators ($H(M_{3b})$ and Σ_{M_3}). The use of $H(M_{3b})$ in Σ_{M_3} allows the verification of M_{3a} without needing M_{3b}, which decreases the overhead caused by the REQUEST packet. The same technique is used in creating Σ_{M_4}.

To limit the number of REQUESTs that traverse an attacker, we exploit the fact that legitimate nodes forward only one REQUEST in any Discovery. First, we require that each REQUEST carry a list of nodes traversed by this REQUEST. Second, we require a bidirectional Neighbor Verification for each link represented by this list of nodes, for a total of two signed Neighbor Verifications per hop. Third, to authenticate the node list, we require each node to authenticate the REQUEST it forwards, though it can piggyback this authentication together with the Neighbor Verification that it signs. Finally, we require buffered REQUESTs be duplicate-suppression-unique: that is, if the route record of any two REQUESTs contain any node A, the route prefix leading up to (and including) A must be the same. These three requirements constrain an attacker to the extent that an attacker that has compromised m nodes can rush at most m REQUESTs.

To prevent replay of old Neighbor Verification messages, each message is tied to a specific Route Discovery. Specifically, when a node S sends a Neighbor Verification for the link from S to R, S signs not just S and R (as in Figure 3), but also ties a unique Route Discovery identifier to the Neighbor Verification. For example, in AODV, the RREQ ID and Originator IP Address in an RREQ form a unique identifier; in DSR, the Target Address and Identifier fields from a ROUTE REQUEST, together with the IP Source Address, form a unique identifier. To address wraparound in these Identifier fields, if the nodes in the network have very loosely synchronized clocks (within a few days), the node can include a timeout in addition to this unique identifier. If network nodes have more tightly synchronized clocks (within a few seconds), the node can include a timeout in place of any unique identifier.

In some areas of some networks, a node will not have n distinct paths to the source of the REQUEST. To enable the Discovery of routes to or through such nodes, we allow a node to forward a REQUEST after some time, even if it has not yet received n REQUESTs. In certain cases, however, a fixed timeout allows an attacker to prevent the discovery of a correct route. One way to avoid such an attack is to choose a random timeout between t_{min} and t_{max}. Alternatively, we can prefer early release when a node has buffered more REQUESTs, for example by choosing a random timeout between $t_{\text{min}} + (n-j)t_{\text{add}}$ and $t_{\text{max}} + (n-j)t_{\text{add}}$, where j is the number of REQUESTs buffered so far. Choosing a timeout when location information is available can provide better properties. If the initiator of each REQUEST includes a timestamp t and its location, intermediate nodes can choose a timeout of t + fixed timeout + propagation speed $\times$ distance to initiator. After a node chooses a timeout, either randomly or based on optional location information, the node randomly chooses one received REQUEST for forwarding.

We implement two additional security optimizations to this basic scheme. In general, these optimizations are based on using the property of nonrepudiation to spread information about malicious nodes. First, we require that each REQUEST be signed by the forwarding node. A node detecting an attacker forwarding more than one REQUEST can expose the attacker by flooding the two REQUESTs. Second, if location information is available, and used for example to implement geographic packet leashes, an attacker claiming to be in two places at the same time can be blacklisted in the same way. For example, if each REQUEST includes in the node list location information and time information for each forwarding node, a node can keep a database of previous location information, and find two location claims that significantly exceed the maximum speed achievable by legitimate nodes. In particular, if location information is accurate to δ, and time information is consistent to Δ, and maximum speed is v, then two locations claimed t time apart is maliciously claimed if the distance between the two locations is greater than $2\delta + v(t + 2\Delta)$. Our blacklist mechanisms do not need authentication, since the nonrepudiation of contradicting information can be can be verified by any nodes. We route blacklist information by flooding: contradictory information is rebroadcast by any node that verifies the nonrepudiation and did not have this malicious node on its blacklist. This approach is similar to the blacklist mechanism used by Ariadne [17].

4.6. Integrating Secure Route Discovery with DSR

To integrate rushing prevention with DSR [19] or other secure protocols based on DSR, we limit Route Discovery frequency as in Ariadne [17]. Each time a node forwards a ROUTE REQUEST, it first performs a Secure Neighbor Detection exchange with the previous hop. When it forwards the REQUEST, it includes in the REQUEST a bidirectional Neighbor Verification for the previous hop.

As in DSR, the target of a Route Discovery returns a ROUTE REPLY for each distinct ROUTE REQUEST it receives. Each such ROUTE REPLY is sent with a source route selected by reversing the route in the ROUTE REQUEST. This route is likely to work if there are no attackers on the route, since Neighbor Detection only finds bidirectional neighbors.

4.7. Integrating Secure Route Discovery with AODV

In AODV [34], as well as other secure protocols based on AODV [7, 40, 46], Route Request (RREQ) packets do not carry a node list. However, in order to filter excessive malicious RREQs, we require each RREQ to carry a node list. Instead of forwarding the first RREQ received, nodes using our Secure Route Discovery randomly select one of the first n RREQs it receives and treats it as the RREQ to forward. More specifically, it places the initiator of the Route

Discovery in its routing table using the previous hop of the RREQ selected as the next-hop destination. It then appends its address and authentication information to the node list, and forwards it as in DSR.

Since AODV is a distance-vector protocol, it cannot make use of multiple routes. As a result, the target of a Route Discovery also waits for *n* RREQ packets before returning a single RREP. The target signs the RREP, and includes in the RREP neighbor authentication for each hop in the chosen path. This authentication allows nodes forwarding the RREP to authenticate the entire path back to the source of the RREP. Each node authenticating this information establishes a route back to the source of the RREP (the target of the RREQ). When this RREP reaches the destination, it will have established a bidirectional route between the initiator and target of the Route Discovery.

Because AODV does not support multiple routes, the security properties of AODV using Secure Neighbor Discovery will be somewhat worse than the properties of DSR using Secure Neighbor Discovery.

4.8. Integrating Secure Route Discovery with Secure Ad Hoc Network Routing Protocols

When using our rushing attack prevention together with a secure on-demand routing protocol, a node can first attempt Route Discovery using that secure protocol. If a rushing attacker prevents the discovery of any working routes, the node can then set a flag indicating that it wants to use rushing attack prevention, though it must also authenticate that flag to prevent modification. This approach is similar to the principle of *expanding ring search*: first, a node uses a cheaper, but sometimes unsuccessful, search. The node only uses a more expensive search when the cheaper search does not find a route. This optimization provides benefits in two cases: first, when there are no rushing attackers, existing secure routing protocols should be able to find a route. Secondly, a rushing attacker does not have any advantage in one- and two-hop routes.

5. EVALUATION

To evaluate our techniques, we analyzed the cost and effectiveness through simulation and analysis. Our simulation was designed to show the cost of our techniques in a non-adversarial environment, whereas our analytical evaluation shows provable bounds on the extent to which an attacker can disrupt a protocol using our techniques.

5.1. Simulation Evaluation

To evaluate the overhead of using our secure neighbor discovery mechanism in a non-adversarial environment, we simulated our scheme using the *ns-2* simulator, using Ariadne [17] as our underlying routing protocol. We call this modified protocol RAP (Rushing Attack Prevention). We did not implement the optimizations described in Section 4.8, because our simulations did not include an attacker, so our results would be equivalent to just using Ariadne. We used the original Ariadne source code [30], and modified it to use digital signatures based on HORS and geographical leashes for wormhole protection [15]. We compared our results with Ariadne and DSR in order to determine the added costs of RAP when there are no attackers. However, when a rushing attacker is present, existing on-demand ad hoc network routing protocol would in general be unable to deliver packets over paths longer than two hops (Section 2). RAP, on the other hand, would be able to discover working paths much of the time, and as a result, would generally outperform existing on-demand routing protocols.

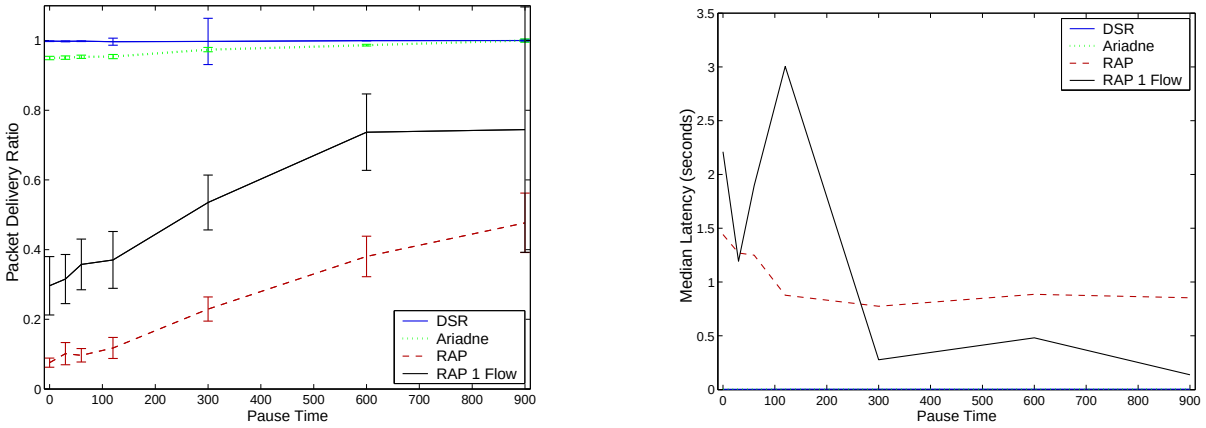
We chose HORS as our broadcast signature, using a time interval of 5 seconds and allowing each node to authenticate up to 20 messages per time interval. We assumed a time synchronization error of 1 second, and used 180 byte signatures. As a result, each public key is 78380 bytes, and each node has an amortized workload of 156760 hash operations per second at each node for generating signatures, as well as verifying all signatures from all nodes. (This level is well within the capability of modern PDAs, and represents around 10% CPU utilization on modern workstations). Our parameters were chosen to provide an 80 bit security level; that is, an attacker must guess 2^{80} signatures to forge one signature in expectation. When signatures were needed at a faster rate than permitted by HORS, we used a multi-signature scheme based on Merkle hash trees [29]. We simulated packet leashes based on optional location information, and waited for 2 REQUEST packets, or a 0.2 seconds fixed timeout plus the distance to the initiator times a propagation speed of 1500 meters per second.

Because a square area is more likely to support multiple routes between a source an a destination, our simulations used 100 nodes in a 1000 m $\times$ 1000 m space moving according to the *random waypoint model* [20]. In this model, each node is randomly placed; at the beginning of the simulation, it waits for a *pause time*, then chooses a velocity uniformly between 0 and 20 meters per second. It then proceeds to a random location at that velocity, and upon arriving, waits for the pause time and repeats. We simulated pause times of 0, 30, 60, 120, 300, 600, and 900 seconds.

We chose a workload of 5 flows, each producing 4 packets per second, using 64-byte packets. This workload was sufficient to cause significant congestion with our scheme, even though normal ad hoc network routing protocols can deliver four or more times the load at lower loss rate; however, secure neighbor discovery incurs significantly higher overhead due to the four-way handshake and speed-of-light delays associated with it. We simulated a link-layer data rate of 2 Mbps.

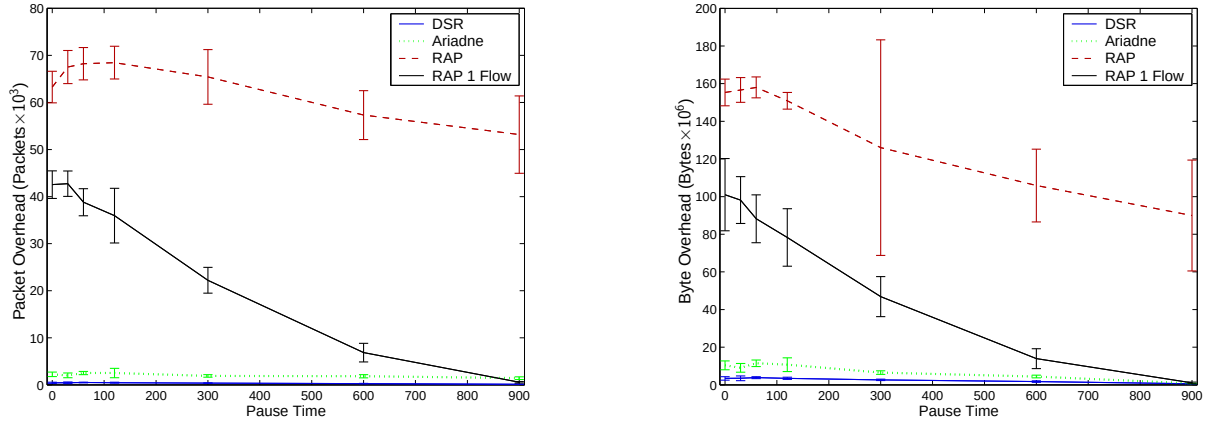
RAP has significantly worse performance than both Ariadne and DSR because of the added load of the Secure Neighbor Discovery. Figure 5(a) shows the Packet Delivery Ratio of the three protocols. DSR delivers between 99.8% and 100% of offered traffic. Ariadne delivers between 95.0% and 100% of offered traffic; a significant improvement over previous simulation results [17]. This suggests that previous simulations used too high a traffic load to fairly evaluate Ariadne in the absence of congestion. Even with this light traffic load, RAP was able to deliver just 7.6% to 47.7% of offered load. This performance is primarily due to congestion. At higher movement speeds (lower pause time), the lower packet delivery ratio is caused by an even higher packet overhead, which results from the on-demand nature of the protocol. We also simulated RAP carrying a lower load of just one flow. At higher pause times, Ariadne with RAP has sufficiently low overhead to deliver between 73.7% and 74.5% of traffic. Even with these pause times, 92.1% of drops were due to MAC-layer congestion, compared to just 4.15% due to the node's inability to find a route. This MAC-layer congestion severely hampers our protocol's ability to deliver application-layer packets.

Figure 5(b) shows the median latency of delivered packets. DSR and Ariadne appear to have zero mean latency, since their median latencies of 4.3ms and 3.8ms respectively are significantly lower than the 1050ms median latency of RAP. Two factors contribute to the higher latency of RAP: first, congestion increases the time each node must wait to acquire the medium, and second, if a node receives just one ROUTE REQUEST packet from a Route Discovery, it waits a significant amount of time before forwarding that REQUEST in an attempt to collect enough REQUESTs and choose one



(a) Packet Delivery Ratio

(b) Median Latency



(c) Packet Overhead

(d) Byte Overhead

Figure 5: Unoptimized RAP performance evaluation results in non-adversarial environment. Optimized RAP would have same results as Ariadne, except that it would perform better when under attack. Under attack, optimized RAP and Ariadne would perform identically for one- and two-hop routes, but in finding longer routes, RAP should significantly outperform Ariadne, since RAP finds working routes with moderate probability, but Ariadne and DSR can never find routes. "RAP 1 Flow" refers to RAP with the lighter communications pattern of one CBR source. Results based on averages over 50 simulation runs; the error bars represent the 95% confidence interval of the mean.

at random.

Figures 5(c) and 5(d) show the Packet Overhead and Byte Overhead of the three protocols. At higher pause times, RAP has more than five times as much overhead when it uses five flows. This indicates that the congestion caused by the protocol significantly reduces the usefulness of the routing protocol packets. When congestion is not an issue, we actually expect that overhead should be less than a factor of five, because nodes can cache information they overhear, thus improving efficiency.

Our performance evaluation shows that in non-adversarial environments, RAP adds significant costs relative to other secure routing protocols. Many of these costs are due to the congestion created at lower bit rates. However, RAP is designed to be used only when necessary (Section 4.8), so these higher costs are only incurred when the underlying protocol is otherwise unable to discover a working route. Specifically, RAP incurs *no cost* until the underlying protocol is completely prevented from finding a working route. It then allows that protocol to use a higher cost approach to successfully deliver packets even against a rushing attacker. In the next section, we show how RAP performs under a rushing attack, in which DSR and Ariadne would be unable to find routes containing more than three nodes (two hops).

5.2. Security Analysis

This section discusses the security properties achieved with RAP when n distinct routes (both legitimate and attacking) exist between the originator and each other node in the network. (As in Section 4.5, two routes are considered distinct if they end in different nodes.) Since routes are required to end in different nodes, an attacker with access to the keys of m compromised nodes can generate at most m distinct, maliciously injected ROUTE REQUESTs for the purpose of denial-of-service.

To analyze the probability of a node subverting a Route Discovery, we assume that the attacker rushes m distinct REQUESTs to each node in the network. As a result, each node needs only $n-m$ additional distinct REQUESTs. We also suppose that the network

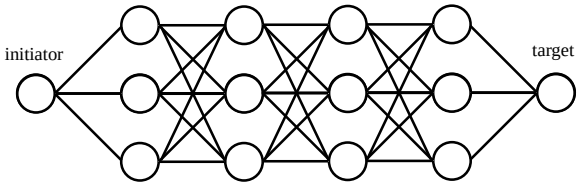


Figure 6: Example network topology used in RAP security analysis.

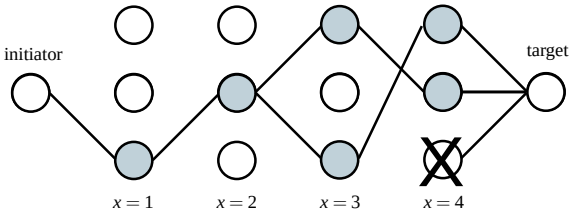


Figure 7: An example of a successful Route Discovery. Each gray node chose a valid REQUEST and belonged to a route for which a REPLY was sent. Each line represents a hop in a path chosen by a legitimate REQUEST; the network topology is shown in Figure 6.

topology of these legitimate requests is represented by Figure 6, such that the ℓ hops from the source to the target form a sequence of tiers, such that the $n-m$ neighbors of the source form the first tier, the $n-m$ neighbors of the target form the last tier, and any two adjacent tiers form a complete, bipartite graph.

We denote the probability of successfully finding a route at tier x given y nodes at that tier to be $S_{x,y}$. In particular, we seek the probability $S_{\ell,n-m}$. Since one-hop neighbors cannot be subverted, $S_{1,y} = 1$ for all $y > 0$. At any other level (that is, when $x \neq 1$), the probability that i of the y neighbors will choose one of the m bogus ROUTE REQUESTs is given by the binomial PDF $\binom{y}{i}\left(\frac{m}{n}\right)^{y-i}\left(\frac{n-m}{n}\right)^{i}$. For example, in Figure 7, at $x = 4$, $y = 3$ nodes received a valid ROUTE REPLY, but only $i = 2$ of them forwarded a valid REQUEST.

Each of the i nodes that do not choose bogus REQUESTs chooses one of the REQUESTs it received. Some of these REQUESTs may overlap; the probability of choosing exactly j distinct previous hops is given by $p_{n-m-j}(n-m,i)$, where $p_{r-j}(r,i)$ is the probability that when i balls are thrown into r boxes, exactly $r-j$ boxes are empty (that is, exactly j boxes are full). The solution to the classical occupancy problem [44] gives $p_{r-j}(r,i) = \binom{r}{r-j}\sum_{k=0}^{j}(-1)^k\binom{j}{k}\left(\frac{j-k}{r}\right)^i$. For example, in Figure 7, at $x = 4$, $i = 2$ nodes chose $j = 2$ distinct previous hops, and at $x = 3$, $i = 2$ nodes chose $j = 1$ distinct previous hops.

When, at a level $x \neq 1$, i nodes do not choose bogus REQUESTs but instead choose a total of j distinct, legitimate REQUESTs, the probability that the Route Discovery will be successful is $S_{x-1,j}$ by definition. Then $S_{x,y}$ is given by the equation in Figure 5.2. For example, when $n = 6$, $m = 2$ and $\ell = 5$, the probability of a successful Route Discovery is 46%.

We now argue that the case above reflects a worst case analysis by analyzing some potential variations. First, the $n-m$ additional incoming nodes could come from earlier tiers (e.g., tiers with lower x). However, since $S_{x,y}$ is monotone decreasing with increasing x and fixed y, the opportunity to choose nodes from earlier tiers only provides a benefit. Second, there may not be as much overlap between the predecessors of the nodes in a single tier; however, this only reduces the number of collisions at the previous tier. Fewer collisions at the previous tier improves performance, since $S_{x,y}$ is monotone increasing with fixed x and increasing y. Third, an attacker can choose to reduce the number of bogus REQUESTs it sends to each node; this has the effect of reducing m, which again increases the probability of success. A final attack allows a powerful attacker to monitor the REQUESTs forwarded by each node legitimate node. Some of these legitimate nodes will have randomly chosen REQUESTs that represent compromised routes. The attacker can then attempt to forward such REQUESTs to nodes that did not hear that REQUEST directly from that node. This attack will be prevented by wormhole detection.

As mentioned in Section 4.7, if only one ROUTE REPLY is returned with any discovery, security is somewhat lower. In particular, only one route is returned, and each hop after the first has a $\frac{n-m}{n}$ probability of choosing a nonattacking node under the attacker model used in this section. In a working route, all nodes must forward a nonattacking REQUEST. As a result, the probability of choosing a working route is $\left(\frac{n-m}{n}\right)^{\ell}$, where ℓ is the number of intermediate nodes (excluding the initator and target).

This section presented an extremely conservative security analysis. In particular, an attacker as aggressive as the one described here would need to propagate the ROUTE REQUEST from each Route Discovery from many different locations, possibly subjecting it to an intrusion detection mechanism. A real attacker considering the tradeoff between an improved probability of subversion and an increased probability of being caught is unlikely to use such a powerful attack.

6. RELATED WORK

We have already discussed the vulnerability of current secure on-demand ad hoc network routing protocols [7, 40, 17, 32, 46] to the rushing attack in Section 2. Perlman's Flooding NPBR [35] routing protocol for wired networks does not suffer from this attack, since the protocol does not depend on the actual path of the flood for routing; rather, it requires that each packet be flooded through the the network.

Other secure routing protocols have been proposed based on periodic (proactive) mechanisms, for wired networks [8, 12, 13, 22, 26, 41, 42] as well as for wireless ad hoc networks [14, 37]. Although these protocols typically are not vulnerable to rushing attacks, such periodic protocols are often less desirable for ad hoc network routing due to their higher overhead and slower adaptivity.

Other areas in secure ad hoc network routing have been explored, such as trust establishment [2, 17, 18, 43], key generation [3], nodes that maliciously do not forward packets [28], and security requirements for forwarding nodes [45]. These areas are beyond the scope of this paper.

Routing protocol intrusion detection has been studied in wired networks as a mechanism for detecting misbehaving routers. Cheung and Levitt [9] and Bradley et al [5] propose intrusion detection techniques for detecting and identifying routers that send bogus routing update messages. In this paper, we describe one invariant of legitimate node behavior, and introduce a distributed mechanism to exclude nodes that have been caught violating that invariant.

7. CONCLUSION

In this paper, we have described the *rushing attack*, a novel and powerful attack against on-demand ad hoc network routing protocols. This attack allows an attacker to mount a denial-of-service attack against *all* previously proposed secure on-demand ad hoc

$$
\begin{aligned}
S_{x,y} &= \sum_{i=1}^{y} \left[\binom{y}{i} \left(\frac{m}{n}\right)^{y-i} \left(\frac{n-m}{n}\right)^{i} \sum_{j=1}^{\min(i,n-m)} S_{x-1,j}\, p_{n-m-j}(n-m,i) \right] \\
&= \sum_{i=1}^{y} \left[\binom{y}{i} \left(\frac{m}{n}\right)^{y-i} \left(\frac{n-m}{n}\right)^{i} \sum_{j=1}^{\min(i,n-m)} \left\{ S_{x-1,j} \binom{n-m}{n-m-j} \sum_{k=0}^{j} (-1)^k \binom{j}{k} \left(\frac{j-k}{n-m}\right)^{i} \right\} \right]
\end{aligned}
$$

Figure 8: The probability of a successful Route Discovery in a network using RAP

network routing protocols. We have also presented RAP (Rushing Attack Prevention), a new protocol that thwarts the rushing attack.

We found that the widely used duplicate suppression technique makes the rushing attack possible, and we designed a new Route Discovery protocol called RAP that replaces the standard mechanism and thwarts the rushing attack. Our approach is generic, so any protocol that relies on duplicate suppression in Route Discovery can use our results to fend off rushing attacks. More importantly, we demonstrated that there are mechanisms that can defend against the rushing attack, even though all previous attempts at secure on-demand ad hoc network routing protocols have been vulnerable.

When integrated with a secure routing protocol, RAP incurs *no cost* unless the underlying secure protocol cannot find valid routes. When RAP is enabled, it incurs higher overhead than do standard Route Discovery techniques, but it can find usable routes when other protocols cannot, thus allowing successful routing and packet delivery when other protocols may fail entirely. We have also shown that existing on-demand routing protocols can be retrofitted using our technique to resist the rushing attack.

REFERENCES

[1] Norman Abramson. The ALOHA System—Another Alternative for Computer Communications. In *Proceedings of the Fall 1970 AFIPS Computer Conference*, pages 281–285, November 1970.

[2] Dirk Balfanz, D. K. Smetters, Paul Stewart, and H. Chi Wong. Talking To Strangers: Authentication in Ad-Hoc Wireless Networks. In *Symposium on Network and Distributed Systems Security (NDSS 2002)*, February 2002.

[3] Stefano Basagni, Kris Herrin, Emilia Rosti, and Danilo Bruschi. Secure Pebblenets. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, pages 156–163, Long Beach, California, USA, October 2001.

[4] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. HMAC: Keyed-Hashing for Message Authentication. Internet Request for Comment RFC 2104, Internet Engineering Task Force, February 1997.

[5] Kirk A. Bradley, Steven Cheung, Nick Puketza, Biswanath Mukherjee, and Ronald A. Olsson. Detecting Disruptive Routers: A Distributed Network Monitoring Approach. In *Proceedings of the IEEE Symposium on Research in Security and Privacy*, pages 115–124, May 1998.

[6] Stefan Brands and David Chaum. Distance-Bounding Protocols. In *Workshop on the theory and application of cryptographic techniques on Advances in cryptology — CRYPTO 1994*, volume 839 of *Lecture Notes in Computer Science*, pages 344–359. Springer-Verlag, August 1994.

[7] Claude Castelluccia and Gabriel Montenegro. Protecting AODV against Impersonation attacks. IETF MANET Mailing List, Message-ID 006601c1eb8c$f279c560$1d1fc7c2@charmette.inrialpes.fr, `https://www1.ietf.org/mail-archive/working-groups/manet/current/msg00186.html`, April 2002.

[8] Steven Cheung. An Efficient Message Authentication Scheme for Link State Routing. In *13th Annual Computer Security Applications Conference*, 1997.

[9] Steven Cheung and Karl Levitt. Protecting Routing Infrastructures from Denial of Service Using Cooperative Intrusion Detection. In *The 1997 New Security Paradigms Workshop*, September 1998.

[10] Thomas Clausen, Philippe Jacquet, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, and Laurent Viennot. Optimized Link State Routing Protocol. Internet-Draft, draft-ietf-manet-olsr-06.txt, September 2001. Work in progress.

[11] Piyush Gupta and P. R. Kumar. The capacity of wireless networks. *IEEE Transactions on Information Theory*, 46(2):388–404, March 2000.

[12] Ralf Hauser, Antoni Przygienda, and Gene Tsudik. Reducing the Cost of Security in Link State Routing. In *Symposium on Network and Distributed Systems Security (NDSS'97)*, pages 93–99, February 1997.

[13] Andy Heffernan. Protection of BGP Sessions via the TCP MD5 Signature Option. RFC 2385, August 1998.

[14] Yih-Chun Hu, David B. Johnson, and Adrian Perrig. Secure Efficient Distance Vector Routing in Mobile Wireless Ad Hoc Networks. In *Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '02)*, June 2002.

[15] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks. In *Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, April 2003. To appear.

[16] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Packet Leashes: A Defense against Wormhole Attacks in Wireless Ad Hoc Networks. In *Proceedings of IEEE INFOCOM 2003*, April 2003.

[17] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A Secure On-Demand Routing Protocol for Ad Hoc Networks. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom 2002)*, pages 12–23, September 2002.

[18] Jean-Pierre Hubaux, Levente Buttyán, and Srdjan Čapkun. The Quest for Security in Mobile Ad Hoc Networks. In *Proceedings of the Third ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, Long Beach, CA, USA, October 2001.

[19] David B. Johnson. Routing in Ad Hoc Networks of Mobile Hosts. In *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94)*, pages 158–163, December 1994.

[20] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In *Mobile Computing*, edited by Tomasz Imielinski and Hank Korth, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[21] David B. Johnson, David A. Maltz, Yih-Chun Hu, and Jorjeta G. Jetcheva. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). Internet-Draft, draft-ietf-manet-dsr-07.txt, February 2002. Work in progress.

[22] Stephen Kent, Charles Lynn, Joanne Mikkelson, and Karen Seo. Secure Border Gateway Protocol (S-BGP) — Real World Performance and Deployment Issues. In *Symposium on Network and Distributed Systems Security (NDSS'00)*, pages 103–116, February 2000.

[23] Stephen Kent, Charles Lynn, and Karen Seo. Secure Border Gateway Protocol (S-BGP). *IEEE Journal on Selected Areas in Communications*, 18(4):582–592, April 2000.

[24] Young-Bae Ko and Nitin Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proceedings of the Fourth International Conference on Mobile Computing and Networking (MobiCom'98)*, pages 66–75, October 1998.

[25] Jiejun Kong, Petros Zerfos, Haiyun Luo, Songwu Lu, and Lixia Zhang. Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks. In *9th International Conference on Network Protocols (ICNP'01)*, 2001.

[26] Brijesh Kumar. Integration of Security in Network Routing Protocols. *SIGSAC Review*, 11(2):18–25, 1993.

[27] Ratul Mahajan, David Wetherall, and Tom Anderson. Understanding BGP Misconfiguration. In *ACM SIGCOMM 2000*, August 2002.

[28] Sergio Marti, T.J. Giuli, Kevin Lai, and Mary Baker. Mitigating Routing Misbehaviour in Mobile Ad Hoc Networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom 2000)*, pages 255–265, Boston MA, USA, August 2000.

[29] Ralph Merkle. Protocols for Public Key Cryptosystems. In *1980 IEEE Symposium on Security and Privacy*, 1980.

[30] The Monarch Project. Rice Monarch Project: Mobile Networking Architectures, project home page. Available at http://www.monarch.cs.rice.edu/.

[31] Richard G. Ogier, Fred L. Templin, Bhargav Bellur, and Mark G. Lewis. Topology Broadcast Based on Reverse-Path Forwarding (TBRPF). Internet-Draft, draft-ietf-manet-tbrpf-05.txt, March 2002. Work in progress.

[32] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure Routing for Mobile Ad Hoc Networks. In *SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, January 2002.

[33] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das. Ad Hoc On Demand Distance Vector (AODV) Routing. Internet-Draft, draft-ietf-manet-

aodv-10.txt, January 2002. Work in progress.

[34] Charles E. Perkins and Elizabeth M. Royer. Ad-Hoc On-Demand Distance Vector Routing. In *Proceedings of the Second IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, pages 90–100, February 1999.

[35] Radia Perlman. *Interconnections: Bridges and Routers*. Addison-Wesley, 1992.

[36] Adrian Perrig. The BiBa One-Time Signature and Broadcast Authentication Protocol. In *Proceedings of the Eighth ACM Conference on Computer and Communications Security (CCS-8)*, pages 28–37, Philadelphia PA, USA, November 2001.

[37] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. SPINS: Security Protocols for Sensor Networks. In *Seventh Annual ACM International Conference on Mobile Computing and Networks (MobiCom 2001)*, Rome, Italy, July 2001.

[38] Raymond L. Pickholtz, Donald L. Schilling, and Laurence B. Milstein. Theory of Spread Spectrum Communications — A Tutorial. *IEEE Transactions on Communications*, 30(5):855–884, May 1982.

[39] Leonid Reyzin and Natan Reyzin. Better than Biba: Short One-Time Signatures with Fast Signing and Verifying. In *Seventh Australasian Conference on Information Security and Privacy (ACISP 2002)*, July 2002.

[40] Kimaya Sanzgiri, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth Belding-Royer. A Secure Routing Protocol for Ad hoc Networks. In *Proceedings of the 10th IEEE International Conference on Network Protocols (ICNP '02)*, November 2002.

[41] Bradley R. Smith and J.J. Garcia-Luna-Aceves. Securing the Border Gateway Routing Protocol. In *Global Internet'96*, London, UK, November 1996.

[42] Bradley R. Smith, Shree Murthy, and J.J. Garcia-Luna-Aceves. Securing Distance Vector Routing Protocols. In *Symposium on Network and Distributed Systems Security (NDSS'97)*, February 1997.

[43] Frank Stajano and Ross Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In *Security Protocols, 7th International Workshop*, edited by B. Christianson, B. Crispo, and M. Roe. Springer Verlag Berlin Heidelberg, 1999.

[44] Richard von Mises. Über Aufteilungs- und Besetzungswahrscheinlichkeiten. *Revue de la Faculté des Sciences de l'Université d'Istanbul*, 4:145—163, 1939.

[45] Seung Yi, Prasad Naldurg, and Robin Kravets. Security-Aware Ad-Hoc Routing for Wireless Networks. Technical Report UIUCDCS-R-2001-2241, Department of Computer Science, University of Illinois at Urbana-Champaign, August 2001.

[46] Manel Guerrero Zapata and N. Asokan. Securing Ad Hoc Routing Protocols. In *Proceedings of the ACM Workshop on Wireless Security (WiSe 2002)*, September 2002.

[47] Lidong Zhou and Zygmunt J. Haas. Securing Ad Hoc Networks. *IEEE Network Magazine*, 13(6), November/December 1999.