Securing Quality-of-Service Route Discovery in On-Demand Routing for Ad Hoc Networks

Yih-Chun Hu
UC Berkeley
yihchun@cs.cmu.edu

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

ABSTRACT

An ad hoc network is a collection of computers (nodes) that cooperate to forward packets for each other over a multihop wireless network. Users of such networks may wish to use demanding applications such as videoconferencing, Voice over IP, and streaming media when they are connected through an ad hoc network. Because overprovisioning, a common technique in wired networks, is often impractical in wireless networks for reasons such as power, cost, and government regulation, Quality of Service (QoS) routing is even more important in wireless networks than in wired networks. Though a number of QoS-routing protocols have been proposed for use in ad hoc networks, security of such protocols has not been considered.

In this paper, we develop SQoS, a secure form of QoS-Guided Route Discovery for on-demand ad hoc network routing. SQoS relies entirely on *symmetric* cryptography. Symmetric cryptographic primitives are three to four orders of magnitude faster (in computation time) than asymmetric cryptography. In addition, we show that in general, existing QoS-Guided Route Discovery can, for a single Route Discovery, transmit a number of packets exponential in the number of network nodes, creating an opportunity for Denial-of-Service (DoS) attacks. SQoS limits this overhead to be linear in the number of network nodes by providing the source with control over which Route Requests are forwarded.

Categories and Subject Descriptors

C.2.0 [**General**]: Security and protection (e.g., firewalls); C.2.2 [**Network Protocols**]: Routing protocols

General Terms

Performance, Design, Security

Keywords

Simulations, Security, Quality-of-Service, QoS Routing, ad hoc networks, SQoS

1. INTRODUCTION

An ad hoc network is a collection of computers (nodes) that cooperate to forward packets for each other over a multihop wireless network. The nodes in the network may move and radio propagation conditions may change at any time, creating a dynamic, rapidly changing network topology. An important application of ad hoc networking technology is to enable communication in environments in which there is no infrastructure, where the infrastructure has been destroyed, or when the infrastructure cannot be used due to issues such as cost and security. A substantial amount of research has been proposed in the field of ad hoc network routing, and mature protocols such as DSR [20], AODV [31], OLSR [33], and TBRPF [3] have emerged from standards discussions in the Internet Engineering Task Force (IETF), the principle protocol standards development organization for the Internet.

Users of ad hoc networks may wish to use demanding applications such as videoconferencing, Voice over IP, and streaming media when they are connected through an ad hoc network. Quality of Service (QoS) has been an important area of research in wired networks, as researchers have looked for solutions that provide acceptable levels of performance for these types of applications. When QoS routing is available in ad hoc networks, users will experience better performance while using these types of challenging applications.

In wireless networks, QoS routing is even more important. That is, in wired networks, overprovisioning can often be used to reduce the need for sophisticated QoS techniques in all but the most demanding network applications. However, in wireless networks, overprovisioning is often impossible or impractical, due to constraints on radio spectrum and power level, or because of interference or noise within the radio spectrum. As a result, using a QoS routing protocol to carefully choose routing paths with sufficient resources may be the only way to provide sufficient resources in wireless networks for many applications. This is even more important in ad hoc networks due to the frequent changes in routing topology and the need to maximize the use of the shared radio resources over multiple wireless hops. In addition, recent work on route selection [10] shows that measuring link quality can significantly improve network performance even for best-effort traffic.

Most work on QoS routing in ad hoc networks has focused on the Integrated Services model [4], with flows and reservations, rather than on the Differentiated Services [30] model, where certain packets are marked as having priority over unmarked packets. One reason most researchers have chosen this direction is that, in ad hoc networks, capacity and connectivity are quite dynamic. The dynamic nature of ad hoc networks pose a significant challenge to the negotiation of Service Level Agreements (SLA), which limit the amount of marked traffic that any node can introduce.

Routing protocols for ad hoc networks can generally be divided into two categories. A *proactive* (or *periodic*) routing protocol attempts to allow each node using it to always maintain an up-to-date route to each possible destination in the network; the protocol periodically exchanges routing information with other nodes in order to allow new routes to be discovered and existing routes to be modified if they break due to factors such as node mobility and environmental changes. A *reactive* (or *on-demand*) routing protocol only attempts to a discover a route to some destination when it has a packet to route to that destination and does not already know a route there; the protocol caches known routes and uses a flooding-based discovery protocol when a needed route is not found in the cache. For dynamically mobile ad hoc networks, reactive or on-demand routing protocols often outperform proactive or periodic ones, due to their ability to adjust the amount of network overhead created to track the mobility in the network affecting current communication [6, 19].

A number of protocols have been proposed for QoS routing in ad hoc networks [5, 7, 8, 24, 25, 32, 38]. As designed, these protocols are intended for operation in a trusted environment in which all nodes are honest, and they do not consider the disruptions that can be caused by a malicious attacker sending arbitrary (e.g., forged) routing packets. In this paper, we discuss general mechanisms for securing QoS routing in on-demand routing protocols for ad hoc networks, and we apply these mechanisms to create a new secure QoS routing protocol that we call SQoS. The operation of SQoS is based on DSR's QoS-guided Route Discovery [25, 5] technique. We also show how our mechanisms can be applied to the proposed AODV QoS extensions [32].

The rest of this paper is organized as follows. In Section 2, we describe existing work on QoS-Guided Route Discovery. In Section 3, we then present an overview of the cryptographic mechanisms that we use for securing QoS-Guided Route Discovery, and Section 4 describes mechanisms for securing QoS-Guided Route Discovery, including one mechanism (ROUTE REQUEST authentication) that is generally applicable to all secure on-demand routing protocols. In Section 5, we then compare our SQoS protocol with a protocol for securing QoS-Guided Route Discovery using public key cryptography. We review related work in Section 6 and conclude in Section 7.

2. QOS-GUIDED ROUTE DISCOVERY

In an on-demand ad hoc network routing protocol, such as DSR [22, 21] or AODV [31, 32], a node (which we call the *initiator*) can find a route to a destination node (which we call the *target*) by performing a controlled flood of the network. In this Route Discovery procedure, the initiator transmits a ROUTE REQUEST packet, identifying the target to which the route is needed. Each node receiving the ROUTE REQUEST in general retransmits the REQUEST if it has not already forwarded a copy of it; when the target node receives the REQUEST, it returns a ROUTE REPLY to the initiator, listing the route taken by the REQUEST, rather than forwarding the REQUEST. Many optimizations have been defined for this basic Route Discovery scheme to reduce the frequency of performing Route Discovery and to limit the portion of the network over which the ROUTE REQUEST flood must be forwarded.

For each individual Route Discovery attempt, each node that forwards the ROUTE REQUEST does so only for the first REQUEST it receives as part of that Route Discovery, in order to limit the overhead of the flooding operation. In AODV, this technique in effect finds the path which forwards the REQUEST to the target with the lowest latency; in DSR, this technique returns a set of paths such that each strict prefix P of each path is the path that forwarded the REQUEST to the last node of P with lowest latency (the target node in DSR returns a ROUTE REPLY for each copy of the ROUTE REQUEST that it receives).

In both cases, a node wishing to find a route with certain QoS parameters (e.g., with a given maximum latency or a given available bandwidth) may not be able to find a route with sufficient quality. In particular, because low-latency paths are preferred, a node searching for a path along which to route a high-bandwidth flow may not find an appropriate route with either DSR or AODV, even if a route with sufficient bandwidth exists. To allow Route Discovery to discover paths satisfying QoS constraints, Maltz introduced QoS-Guided Route Discovery [5, 25], which allows a node to specify QoS metrics that must be satisfied by a discovered path. (The QoS-Guided Route Discovery technique was specified in version 3 of the DSR Internet-Draft [5] in the IETF but was removed in later drafts [22] to simplify the standard DSR protocol specification in accordance with the consensus of the MANET Working Group.) In this section, we review the previous work in QoS-Guided Route Discovery.

In QoS-Guided Route Discovery, ROUTE REQUEST packets are constrained to paths fulfilling certain requirements. Often, a node may already have a preexisting route to the destination; for example, in DSR, a node may have a cached route, or in AODV, a node may already have the destination in its routing table. When a node has a preexisting route, it may either perform a QoS-Guided Route Discovery, or it may attempt to establish a new flow along the preexisting route. If the node chooses to use the preexisting route and the flow establishment is successful, it is not necessary to perform a QoS-Guided Route Discovery, although one may be performed in an attempt to find a better route. The decision about whether or not to perform such a Discovery may be made based on resources available along a preexisting route or the node's estimate of the probability of successful flow setup along that route. Alternatively, a node may choose to always perform a second search requesting a slightly higher level of resources than is available along the preexisting route.

To use this QoS-Guided Route Discovery mechanism, a node sending a ROUTE REQUEST also inserts in the REQUEST an optional *QoS Request Header* for each type of resource required. Each QoS Request Header indicates the type of resource, the minimum acceptable resource level, and the resource level of the current path. The resource level of the current path is initialized to the desired resource level, but may be reduced as the ROUTE REQUEST traverses the network. For example, an audio flow may require at least 2.4 kbps of bandwidth but desire up to 128 kbps. In this case, when it initiates the Route Discovery, the initiator node specifies a minimum acceptable level of 2.4 kbps and a current resource level of 128 kbps.

A node receiving a ROUTE REQUEST containing one or more QoS Request Headers processes each QoS Request Header to determine whether or not the node can support a new flow with resources at a level at least equal to the minimum requested. If it is unable to support the minimum requested resource level for any requested resource, the node silently discards the ROUTE REQUEST. If it is unable to support the current level specified in any QoS Request Header in the packet, the node modifies the header by setting the current level equal to the maximum resource level it can support, and then forwards the ROUTE REQUEST normally. A node able to support the current level specified in all QoS Request Headers contained in the packet forwards the ROUTE REQUEST packet normally without modifying the QoS Request Header.

We now consider the three common QoS metrics of *bandwidth*, *latency*, and *jitter*. With the bandwidth metric, a node forwarding a packet places in the current resource level the lesser of the resource level that it received and its own resource level. When a node with 240 kbps of available bandwidth receives a REQUEST with a current resource level of 640 kbps, it reduces the bandwidth level in the REQUEST before forwarding it. For the metrics of latency and jitter, each node actually increases the latency and jitter specified in the REQUEST, and therefore adds the local latency or jitter to the received value. For example, if a node receives a REQUEST reflecting 20 ms of latency and 5 ms of jitter, and the node itself imposes 10 ms of latency with 3 ms of jitter, the REQUEST packet it forwards will show 30 ms of latency and 8 ms of jitter.

The QoS Request Headers in a ROUTE REQUEST only determine if the requested resources are available along the path, limiting the Route Discovery to return only paths that meet at least the minimum levels of resources requested. A node that propagates a ROUTE REQUEST containing QoS Request Headers may also temporarily reserve the resources specified in the REQUEST in order to improve the likelihood that the resources will still be available when the flow begins using this route.

One important problem in QoS-Guided Route Discovery is determining the resources available at any particular node. These techniques are beyond the scope of this paper, but some earlier work has addressed this problem. For example, Maltz describes techniques for measuring latency and available bandwidth [25]. Any technique used to measure available performance may themselves be manipulated by an attacker. In this paper, we assume that such attacks can only reduce the measured available resources, and, in doing so, reduce actual available resources. For example, an attacker may reduce apparent bandwidth by unnecessarily reserving the medium through the use of excessive RTS and CTS packets [18, 2]; however, this attack actually does reduce available bandwidth.

A routing protocol using QoS-Guided Route Discovery can find suitable routes through the network. Once such a route is found, the routing protocol either must reserve those resources for a flow, or it will use that route on a best effort basis. For example, when the route is used on a best effort basis, a source might use the route until performance degrades to an unacceptable level, at which point it would re-initiate QoS-Guided Route Discovery. Alternatively, a protocol may allow a path establishment and resource reservation protocol, in which a source establishes a flow along that path by sending an ESTABLISH FLOW packet along that path [16]. Each node along the path receiving the ESTABLISH FLOW packet reserves the resources needed by the flow and forwards the ESTABLISH FLOW packet to the next node on the path. When a node that has been forwarding traffic for a flow is no longer able to meet the QoS requirements of the flow, it sends a FLOW ERROR packet to the source of the flow.

Though flow establishment requires two additional packet types, they are specific to the routing protocol in use. In general, ESTABLISH FLOW packets can be authenticated either through broadcast authentication (e.g., as described in Section 4.1), or through the use of pairwise authentication using shared keys between the source and each forwarding node. One of these two types of authentication is generally required to secure other routing protocol messages. When each forwarding node can authenticate the source, it can use policy to determine whether or not that source is authorized to reserve these resources. In addition, FLOW ERROR packets can be authenticated in the same way as ROUTE ERROR packets used by on-demand routing protocols. For example, when used with Ariadne using digital signatures, the ESTABLISH FLOW packet can be digitally signed by the source, so each forwarding node can ensure that the source is authorized to make that reservation; likewise, each FLOW ERROR packet is digitally signed by the node originating the ERROR, so the source can ensure that the ERROR was in fact sent by a node on the route. Because each routing protocol has different key setup requirements and secures routing messages differently, we leave the security of these control messages to the routing protocol; in the rest of the paper, we focus on securing the QoS-Guided Route Discovery mechanism itself.

3. CRYPTOGRAPHIC MECHANISMS

We design SQoS, our secure QoS routing protocol, by building on existing security mechanisms. Specifically, SQoS builds on hash chains and MW-chains, which we review in this section.

3.1 HASH CHAINS

One-way hash chains are a widely used cryptographic primitive. One of the first uses of one-way chains was in one-time password protocols [23, 14]. These chains are also used in other applications, such as efficient one-time signature algorithms [12, 28, 27, 35]. Coppersmith and Jakobsson present efficient mechanisms for storing and generating values of hash chains [9].

We create a one-way chain by selecting the final value v_n at random, and by repeatedly applying a one-way hash function H, such that $v_i = H[v_{i+1}]$. The last value generated in this way is called the *anchor*; generally, an authentic anchor is published to allow verification of hash chain elements. One-way chains have two main properties (assuming H is a cryptographically secure one-way hash function):

- Anybody can authenticate that a value v_j really belongs to the one-way chain, by using an earlier value v_i of the chain by checking that $H^{j-i}(v_j)$ equals v_i.
- Given the latest released value v_i of a one-way chain, an adversary cannot find a later value v_j such that $H^{j-i}(v_j)$ equals v_i. Even when value v_{i+1} is released, a *second pre-image collision resistant hash function* prevents an adversary from finding v'_{i+1} different from v_{i+1} such that $H[v'_{i+1}]$ equals v_i.

These two properties result in authentication of one-way chain values: if the current value v_i belongs to the one-way chain, and we see another value v_j with the property that $H^{j-i}(v_j)$ equals v_i, then v_j also originates from the same chain and was released by the creator of the chain.

3.2 THE MW CHAINS MECHANISM

In this section, we review the MW-chain mechanism [15], which provides instant authentication and low storage overhead. First, we describe the one-time signature, on which MW-chains are based. In a signature, a node chooses a private key K, and from that private key generates a verification key V. Given a message m, the node can use K to form a signature s such that a node with V can verify the signature; however, a node with V but not K cannot generate a signature. A *one-time* signature is a type of signature such that only one message m can be signed with a single key. For example, in the Merkle-Winternitz one-time signature, two signatures using the same key provide an attacker enough information to forge certain other signatures.

The MW-chain is built on a certain type of one-time signature, which we call a *chainable* signature. In a chainable signature, a signature s on message m can be verified by comparing $f(s,m)$ to verification key V, and any verification key can be used as a signature key. One such one-time signature is the Merkle-Winternitz signature.

To build an MW-chain of length ℓ from a chainable signature, we pick a signing key K_ℓ. We then derive each signing key K_i as the public key corresponding to signing key K_{i+1}. In particular, if function G generates a verification key from a signature key, then $K_i = G[K_{i+1}]$. Since G must be a secure one-way hash function, an MW-chain has the same properties of a hash chain, and has the additional property that a signature s using key K_{i+1} can be used to generate K_i using the equation $K_i = f(s,m)$, but cannot be used to derive K_{i+1}.

4. MECHANISMS FOR SECURING QOS ROUTING

Our key observation for securing QoS routing is that properties of interest in the route discovery and selection are generally monotone; that is, the desirability of a path decreases as more nodes are added. For example, the resources of bandwidth, latency, and jitter all are monotone. In this paper, we design SQoS, a secure QoS routing protocol that enforces monotonicity and strict monotonicity in QoS metrics, preventing an attacker from subverting the correct operation of the QoS routing.

4.1 BROADCAST AUTHENTICATION FOR ROUTE REQUEST PACKETS

Our mechanisms for secure QoS Route Discovery require the network to provide some form of *broadcast authentication* for the immutable fields of ROUTE REQUEST packets; that is, *any* node that receives a ROUTE REQUEST packet must be able to ascertain that it was sent by the claimed initiator. Though this authentication can be provided by a digital signature, the cost of verifying a digital signature creates the possibility of a Denial-of-Service (DoS) attack; in this attack, an attacker floods a victim node with invalid ROUTE REQUEST packets, forcing the victim to consume all of its CPU time attempting to check the signatures on the REQUESTs. An alternative to the use of digital signatures is to provide this authentication using an efficient, instant broadcast authentication mechanism such as HORS [34].

In SQoS, however, this authentication is integrated with a mechanism that prevents excessive flooding. In particular, since QoS-Guided Route Discovery requires a flood of the network and hence provides a means for an attacker to perform a Denial-of-Service attack (consuming all network resources), a secure ad hoc network routing protocol must enforce limits on the frequency at which each node can initiate such flooding. For example, Ariadne [18] uses a hash chain to provide this rate-limit. This technique has two advantages. First, it allows any node to authenticate that a ROUTE REQUEST did in fact originate from the initiator. Second, it uses only efficient symmetric cryptography. However, this technique does not prevent modification of the fields of the REQUEST. We replace this hash chain with an MW-chain to prevent the modification of the immutable fields of the REQUEST. A node uses one MW-chain step for each Route Discovery, and uses the signature from that MW-chain step to authenticate the immutable fields of the ROUTE REQUEST.

For example, if the MW-chain allows the authentication of 2^{80} different values, then an 80-bit one-way hash of the immutable fields of the packet can be encoded as a single value authenticated using this MW-chain. An attacker attempting to change any or all of these immutable fields, then, will have a 2^{-80} probability of preserving the correctness of the signature.

4.2 ENFORCING MONOTONICITY

To ensure monotonicity, the initiator of a QoS-Guided Route Discovery creates a virtual hash chain for each QoS metric requested. This virtual hash chain can be a traditional hash chain, as described in Section 3.1, a skipchain [15] (which allows for more efficient authentication of large changes in metric), or a hash tree chain [15] (which enforce strict monotonicity, effectively requiring each forwarder to change the QoS metric). In the rest of this section, for simplicity, we will describe SQoS using a traditional hash chain; however, SQoS can also be used with these other types of one-way chains.

To generate this hash chain, the initial value (value farthest from the anchor) is chosen to authenticate the maximum level of service requested by this QoS-Guided Route Discovery for this metric. Each step in the hash chain authenticates one *quanta*, which is the smallest difference that can be represented for the authentication for that metric. The hash chain is generated to represent each value between the maximum level of service (the initially generated value) and the minimum level of service (the anchor) requested by the QoS-Guided Route Discovery. For example, if an initiator is interested in a range of bandwidth between 2.4 kbps and 56 kbps, and bandwidth is specified in bits per second, the node generates a hash chain of length $56000 - 2400 = 53600$. Each step in this chain represents 1 bps, such that the initially chosen seed represents the 56 kbps level and the anchor represents the 2.4 kbps level. A ROUTE REQUEST packet includes the authenticator for the metric currently claimed in that ROUTE REQUEST. When a node forwarding the REQUEST reduces the metric claimed in the REQUEST, it hashes the authenticator accordingly; for example, if a node reduces the claimed bandwidth by 5 kbps, it applies the one-way hash function 5000 times. Each anchor is included in the Route Discovery, sent with broadcast authentication (as described in Section 4.1), allowing each recipient to authenticate each claimed QoS metric.

The quantization of integer or fixed point values is simpler than for floating point values. For example, the smallest step representable with an integer is 1, and the range of a 32-bit signed integer value is 2^{32}, whereas a 32-bit signed floating point value in the IEEE 754 standard can represent a step as small as 2^{-149} with a range of 2^{129}. We can overcome this difficulty by using a variable step size; since n bits can represent at most 2^n values, we conceptually sort all representable values, and correlate one step in a virtual hash chain with one element in this conceptually sorted list. This sorting can be achieved at low computational cost with proper data representation; for example, finite positive numeric floating point values in the IEEE 754 standard can be converted into their integer ranks by taking the 32-bit (or 64-bit) binary representation and interpreting the opaque bit values as if those bits represented an unsigned integer.

Even though hash functions are relatively fast to compute, following a hash chain of length 2^{32} for each ROUTE REQUEST packet would provide an opportunity for Denial of Service attacks based on flooding REQUESTs with invalid metric authenticators. To reduce the maximum amount of effort needed to verify any single metric, SQoS uses a network-wide maximum number of steps between the minimum level of service required and the maximum level that can be used. Instead of authenticating the exact level of service, only the current step is authenticated, though a more precise measurement is included in the forwarded REQUEST. Since SQoS authenticates only the step (as opposed to the exact metric value), an attacker can change the metric to any value within the same step. For example, for the metric of bandwidth, the span from minimum acceptable to maximum usable could be as great as from 2.4 kbps to 2.4 Mbps for a conferencing application. Divided logarithmically into 200 steps, each step represents a factor of 1.035 increase, and a node claiming a bandwidth of 125 kbps would include an authenticator valid for any bandwidth between 123.65 kbps and 128 kbps. In general, for each metric, the initiator specifies a minimum level acceptable, a maximum level usable, the number of steps between those two levels, and whether those steps are calculated linearly or logarithmically.

To prevent an attack where a node uses maximum-length chains for each of a number of metrics, the network-wide maximum can provide the maximum number of steps summed across all metrics. For example, if a total of 300 steps were allowed, a node may wish to use 200 steps to represent bandwidth with logarithmic steps and 100 steps to represent latency with linear steps.

4.3 LIMITING OVERHEAD OF QOS-GUIDED ROUTE DISCOVERY

In QoS-Guided Route Discovery, a forwarding node does not perform *duplicate suppression* as standard Route Discovery does. In standard Route Discovery, nodes having already forwarded a ROUTE REQUEST from a Route Discovery ignore further REQUESTs from the same Discovery. In QoS-Guided Route Discovery, a node should only ignore a REQUEST if it has forwarded a *better* REQUEST. This raises three problems:

- An intermediate node may not know which tradeoffs between QoS metrics are preferred by the source (e.g., does the source prefer 1 Mbps and 50 ms latency, or 2 Mbps and 75 ms latency?)

- An attacker can force a node to forward a large number of ROUTE REQUESTs by broadcasting a single REQUEST multiple times, using progressively better metrics.

- If a node forwards each better REQUEST, an exponential number of forwarded packets can result from a single Route Discovery.

The first problem exists because different types of traffic demand different link qualities. For example, voice traffic may be very jitter-sensitive, somewhat latency-sensitive, and relatively bandwidth insensitive, whereas a bulk data transfer may be highly bandwidth-sensitive. A strict priority amongst the QoS parameters may be impractical, since videoconferencing software may be able to code video and audio at several quality (and hence bandwidth) levels but also desire low latency and jitter; preferring bandwidth (and hence video quality) over the latency and jitter metrics may result in a path with suboptimal latency and jitter characteristics, whereas optimizing latency and jitter may result in a bandwidth level that requires the use of a lower-quality video codec.

Depending on how intermediate nodes in the node list are authenticated, the second problem may or may not exist. In particular, if a digital signature or other instantly verifiable broadcast authentication is used for node authentication, then each forwarding node can verify that the ROUTE REQUEST has traversed the sequence of nodes listed in the node list. In this case, an attacker can only play as many REQUESTs as it receives, because otherwise the authenticated REQUEST proves that the attacker is behaving maliciously. However, if the node authentication is performed at the target, as in Ariadne [18], an attacker can perform this attack at will.

The third problem, however, is fundamental when forwarding latency of a ROUTE REQUEST packet at a node is not exactly correlated with the initiator's route preference. For example, suppose there are n nodes in addition to the initiator and the target, arranged in $n/2$ groups of 2 nodes as shown in Figure 1. Let group 0 represent the target and group $n/2+1$ represent the initiator. If these groups are arranged such that both nodes in group i are neighbors of all nodes in groups $i-1$, and $i+1$, of the other node in group i, and of no other nodes. Then each node in group 1 would forward 1 REQUEST, for a total of 2 REQUESTs forwarded. If the node in group 1 with the lowest level of resources was the first to forward the REQUEST, then each node in group 2 would forward both REQUESTs, for a total of 4 REQUESTs forwarded. Again, in the worst case, each node in group 3 would forward all 4 REQUESTs, for a total of 8 REQUESTs forwarded. In general, group i would forward 2^i REQUEST packets, for a total number of ROUTE REQUEST packet transmissions of

$$\sum_{i=0}^{n/2} 2^i = 2^{\frac{n}{2}+1} - 1$$

SQoS solves all three problems by providing the source with control over which ROUTE REQUESTs are re-forwarded. For example, a node can include an evaluation function in each REQUEST. This evaluation function can take the form of a function selected from a list, or can be more general, such as active code, as has been proposed for Active Networks [37]. Each evaluation function should take as input the metrics of interest and a maximum value, and should return an integer between 0 and a maximum value specified in the REQUEST. A node then can forward an additional REQUEST only when the evaluation function returns a larger value than it did the previous time, thus allowing each node to bound the number of times it forwards a REQUEST from any single Discovery. To prevent a DoS attack where the attacker allows a REQUEST to be forwarded a large number of times, we can specify as a network-wide parameter the maximum number of REQUESTs that can be forwarded by a single node from a single Route Discovery. Alternatively, we can limit the rate at which REQUESTs can be forwarded for any particular node; since the maximum return value of the evaluation function is authenticated, a node can ignore REQUESTs beyond a certain limit. To prevent an attacker from specifying extremely CPU intensive programs, SQoS uses a language with no loop or subroutine constructs, so that runtime is proportional to program length. Other techniques that can achieve the same result include the use of a "sandbox" that

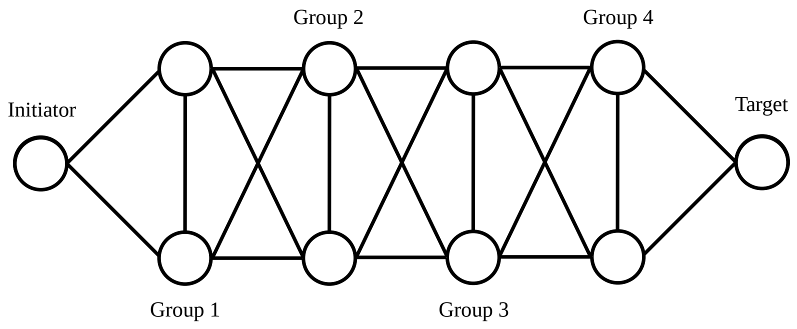

Figure 1: Network topology used in counting ROUTE REQUESTs forwarded by each node

limits resource consumption, or Proof Carrying Code [29] that ensures bounded runtime.

4.4 APPLICATION TO DSR

To perform secure QoS-Guided Route Discovery in on-demand source routed protocols such as DSR [20], and in secure versions of DSR such as Ariadne [18], an initiator using SQoS specifies a list of metrics of interest, such as latency and bandwidth. For each metric, the initiator indicates the maximum necessary level and minimum desirable level, the length of the hash chain, and whether steps are to be divided linearly or logarithmically.

To authenticate these levels, hash chain anchors, and other fields of the ROUTE REQUEST, we add an authentication header to the REQUEST based on an MW-chain; a node's ith Route Discovery is signed using private key K_i, and since Route Discoveries flood the entire network, most nodes will have already heard a signature using the previous key K_{i-1}. As a result, they need only follow one step in the MW-chain to verify the signature, which results in very efficient verification.

For example, in the topology shown in Figure 2, a node S may initiate a ROUTE REQUEST to node D for a route to be used for a videoconference. Node S may want bandwidth between 64 kbps and 1.28 Mbps and may want latency between 0 ms and 200 ms. Node S may allocate 200 logarithmically-divided steps to bandwidth and 100 linearly-divided steps to latency, so each step of bandwidth represents a factor of 1.015, and each step of latency represents an additional 2 ms. It then builds the hash chains for authenticating bandwidth (h_i^B) and for authenticating latency (h_i^L). For bandwidth authenticators, initiator S chooses h_{200}^B and the hash chain using $h_i^B = H[h_{i+1}^B]$; it then chooses latency authenticator h_{100}^L and computes the hash chain $h_i^L = H[h_{i+1}^L]$, as described in Section 4.2. The two anchors h_0^B and h_0^L are included in the REQUEST packet. In addition, the initiator specifies some limit to the number of REQUEST packets forwarded by any node as described in Section 4.3; for example, the initiator may want each node to forward at most 5 REQUESTs from this Route Discovery, and it may equally weight the 200 bandwidth and 100 latency steps. A simple postfix program that would achieve this is `1 LOADSTEP 2 LOADSTEP 2 * + 5 * 400 /`, where `LOADSTEP` is a unary operator that loads current step number of the ith metric, where i is the input value. For example, `1 LOADSTEP` loads the step that the bandwidth is on; if the bandwidth level is 1.28 Mbps, which corresponds to bandwidth step 200, `1 LOADSTEP` results in the value 200 on the stack. The program above adds the bandwidth step number to twice the latency step number (since there are a maximum of 100) and scales the result to a value between 0 and 5. The initiator then signs the REQUEST using its next MW-chain element. It then adds the current path bandwidth (1.28 Mbps) and latency (0 ms) and the authenticators for those values. It also adds a path list, which starts empty. Finally, it broadcasts the resulting ROUTE REQUEST:

$$\begin{aligned} S \rightarrow *: \quad & [\text{ROUTE REQUEST}, D, i, \\ & (\text{Bandwidth}, 64000, 1280000, 200, \text{Log}, h_0^B), \\ & (\text{Latency}, 0, 200, 100, \text{Linear}, h_0^L), \\ & 5, (\texttt{1 LOADSTEP 2 LOADSTEP 2 *} \\ & \quad \texttt{+ 5 * 400 /})]_{K_i^-}, \\ & 1280000, h_{200}^B, 0, h_{100}^L, () \end{aligned}$$

As in DSR, each node keeps a table of ROUTE REQUEST packets it has previously heard. However, unlike in DSR, MW-chain position must be "hard state" to ensure security; that is, if a node forgets its MW-chain position, the correct operation of the protocol is jeopardized. As a result, each node need store only information from the most recent REQUEST initiated by each other node in the network. In addition to the MW-chain position (which replaces the identifier) and initiator, the node stores three additional values: the maximum output of the evaluation function (in this case, 5), the output of the evaluation function for the REQUEST last forwarded from this Route Discovery, and a hash of the immutable (signed) fields of the header. This hash prevents an attacker from forging multiple REQUESTs using an old MW-chain position once the new MW-chain signature has been revealed. It also speeds up the verification of the signature.

When a node receives a REQUEST, it checks if it has received a REQUEST with this or newer MW-chain position and authenticates the signature. If the node has not previously seen a REQUEST with this or newer MW-chain position, it forwards the REQUEST and updates its table of previously heard REQUESTs. If a more recent REQUEST has been heard, this REQUEST is discarded; if a REQUEST from the same Route Discovery has been heard, the node evaluates the new REQUEST according to the evaluation function in the packet. If the evaluation function returns a higher value for this new REQUEST than for the previously forwarded REQUEST, it forwards this REQUEST and also updates the table of previously forwarded REQUESTs. For example, a node hearing the REQUEST above would check if it had heard REQUEST i or larger. If it had not, it would authenticate the signature and evaluate the evaluation function, which would return 5. The node would then

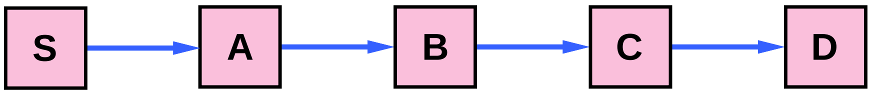

Figure 2: A Simple Example Ad Hoc Network Topology

note that for initiator S it had forwarded REQUEST i with value 5 of a maximum of 5.

When forwarding a ROUTE REQUEST, a node reduces the QoS levels in the REQUEST to the levels that it can provide. For example, a node A which adds 20 ms of latency and has 640 kbps of available bandwidth would reduce the above REQUEST to reflect a 640 kbps bandwidth (based on the maximum of 640 kbps and 1.28 Mbps) and 20 ms latency (based on the sum of 0 ms plus 20 ms), as described in Section 2. It then computes the authenticators for bandwidth and latency h^B_{154} and h^L_{90}, and forwards the ROUTE REQUEST:

$$\begin{aligned} A \rightarrow * : \quad & [\text{ROUTE REQUEST}, D, i, \\ & (\text{Bandwidth}, 64000, 1280000, 200, \text{Log}, h^B_0), \\ & (\text{Latency}, 0, 200, 100, \text{Linear}, h^L_0), \\ & 5, (\texttt{1 LOADSTEP 2 LOADSTEP 2 *} \\ & \quad \texttt{+ 5 * 400 /})]_{K_i^-}, \\ & 640000, h^B_{154}, 20, h^L_{90}, (A) \end{aligned}$$

If node B, which adds 10 ms of latency and has 960 kbps of bandwidth capacity, forwards this ROUTE REQUEST, it does not reduce bandwidth, since it can support this flow. However, it increases latency to 30 ms and computes authenticator h^L_{85}. It also computes the evaluation function, for which it receives a result of 4. It stores this result in the table of forwarded REQUESTS, and forwards the ROUTE REQUEST:

$$\begin{aligned} B \rightarrow * : \quad & [\text{ROUTE REQUEST}, D, i, \\ & (\text{Bandwidth}, 64000, 1280000, 200, \text{Log}, h^B_0), \\ & (\text{Latency}, 0, 200, 100, \text{Linear}, h^L_0), \\ & 5, (\texttt{1 LOADSTEP 2 LOADSTEP 2 *} \\ & \quad \texttt{+ 5 * 400 /})]_{K_i^-}, \\ & 640000, h^B_{154}, 30, h^L_{85}, (A, B) \end{aligned}$$

If node C, which adds 15 ms of latency and has 240 kbps of capacity, forwards this ROUTE REQUEST, it reduces bandwidth to 240 kbps and increases latency to 45 ms. The authenticators for bandwidth and latency become h^B_{89} and h^L_{78}. Node C also computes the evaluation function, for which it receives a result of 4. It stores this result in the table of forwarded REQUESTS, and forwards the ROUTE REQUEST:

$$\begin{aligned} C \rightarrow * : \quad & [\text{ROUTE REQUEST}, D, i, \\ & (\text{Bandwidth}, 64000, 1280000, 200, \text{Log}, h^B_0), \\ & (\text{Latency}, 0, 200, 100, \text{Linear}, h^L_0), \\ & 5, (\texttt{1 LOADSTEP 2 LOADSTEP 2 *} \\ & \quad \texttt{+ 5 * 400 /})]_{K_i^-}, \\ & 240000, h^B_{89}, 45, h^L_{78}, (A, B, C) \end{aligned}$$

Finally, for each ROUTE REQUEST the target receives, it returns a ROUTE REPLY. This REPLY can be authenticated with a key shared between the initiator and the target, or, if no such key exists, the initiator and target can use Diffie-Hellman key exchange (or other key exchange that does not require communication) to establish a key. Since we use MW-chain authentication to rate-limit the number of Route Discoveries initiated by any node, an attacker would need to compromise a large number of nodes to overwhelm a victim with many legitimate REQUESTS for which the victim needs to establish keys. In particular, for the ROUTE REQUEST above, node D returns a ROUTE REPLY:

$$D \rightarrow C : MAC_{K_{SD}}[\text{ROUTE REPLY}, i, 240000, 45, (A, B, C)]$$

Each node forwards the ROUTE REPLY towards the source. A complete execution of the secure QoS-Guided Route Discovery protocol is shown in Figure 3.

4.5 APPLICATION TO AODV

The application of secure QoS-Guided Route Discovery to AODV [31], and to secure versions of AODV such as ARAN [36] and SAODV [39], is similar to the application to DSR and Ariadne. We highlight the differences here.

Since AODV does not maintain a source route in its ROUTE REQUEST packets (called RREQs in AODV), secure QoS-Guided Route Discovery also does not require the source route. As a result, during Route Discovery, downstream nodes do not know the complete route that the RREQ followed, but instead only know the address of the last node to forward the RREQ. Each node forwarding an RREQ keeps a table of the previous hop for each RREQ that it has forwarded. We can use this property to improve routes even after an RREQ is forwarded, by modifying the evaluation function to allow the return of fractional values. An additional RREQ is forwarded for a Route Discovery if the evaluation function applied to the new RREQ is at least 1 greater than for the previously forwarded RREQ. If it is greater but not sufficiently improved, the previous hop can be updated to reflect the new path, but only if each metric of the new RREQ is at least as good as the forwarded RREQ.

When AODV is used as the underlying routing protocol, the initiator of the Route Discovery is not provided with the addresses of each forwarding node. As a result, authentication of a ESTABLISH FLOW packet must be performed using broadcast authentication. Fortunately, the two proposed schemes for securing AODV [39, 36] both use broadcast authentication to secure other protocol messages, so our secure QoS-Guided Route Discovery protocol can be used without any additional key setup.

5. EVALUATION

To evaluate our protocol, we first analyze the security properties it provides. Then, to quantify the costs of our scheme, we define a public-key based secure QoS-Guided Route Discovery mechanism based on prior work in the related area of secure ad hoc network routing. We then compare SQoS to this public-key system, which we call the Public Key Secure QoS Route Discovery (PK-Squared), and show that SQoS significantly outperforms PK-Squared.

5.1 SECURITY ANALYSIS

To analyze the security achieved by our secure QoS-Guided Route Discovery scheme, we examine the taxonomy of attacks

$S \rightarrow *:$ [ROUTE REQUEST, D, i, (Bandwidth, 64000, 1280000, 200, Log, h_0^B), (Latency, 0, 200, 100, Linear, h_0^L), 5, (`1 LOADSTEP 2 LOADSTEP 2 * + 5 * 400 /`)]$_{K_i^-}$, 1280000, h_{200}^B, 0, h_{100}^L, ()

$A \rightarrow *:$ [ROUTE REQUEST, D, i, (Bandwidth, 64000, 1280000, 200, Log, h_0^B), (Latency, 0, 200, 100, Linear, h_0^L), 5, (`1 LOADSTEP 2 LOADSTEP 2 * + 5 * 400 /`)]$_{K_i^-}$, 640000, h_{154}^B, 20, h_{90}^L, (A)

$B \rightarrow *:$ [ROUTE REQUEST, D, i, (Bandwidth, 64000, 1280000, 200, Log, h_0^B), (Latency, 0, 200, 100, Linear, h_0^L), 5, (`1 LOADSTEP 2 LOADSTEP 2 * + 5 * 400 /`)]$_{K_i^-}$, 640000, h_{154}^B, 30, h_{85}^L, (A, B)

$C \rightarrow *:$ [ROUTE REQUEST, D, i, (Bandwidth, 64000, 1280000, 200, Log, h_0^B), (Latency, 0, 200, 100, Linear, h_0^L), 5, (`1 LOADSTEP 2 LOADSTEP 2 * + 5 * 400 /`)]$_{K_i^-}$, 240000, h_{89}^B, 45, h_{78}^L, (A, B, C)

$D \rightarrow C:$ $MAC_{K_{SD}}$[ROUTE REPLY, i, 240000, 45, (A, B, C)]

$C \rightarrow B:$ $MAC_{K_{SD}}$[ROUTE REPLY, i, 240000, 45, (A, B, C)]

$B \rightarrow A:$ $MAC_{K_{SD}}$[ROUTE REPLY, i, 240000, 45, (A, B, C)]

$A \rightarrow S:$ $MAC_{K_{SD}}$[ROUTE REPLY, i, 240000, 45, (A, B, C)]

Figure 3: Secured QoS-Guided Route Discovery

provided by Hu et al [18] and Dahill et al [36]. In our analysis, we consider only the additional risks posed by QoS-Guided Route Discovery, and not those caused by the underlying ad hoc network routing protocol. In particular, because QoS-Guided Route Discovery can be implemented with only one additional packet type (ESTABLISH FLOW), and because the authentication of that packet can generally be performed in the same way as for ROUTE REQUESTs, we ignore general routing attacks such as corrupted node lists, wormhole attacks, routing message replay, spoofed signaling, fabricated routing messages, routing loop formation, and the participation of unauthorized nodes. We focus instead on attacks specific to the QoS-Guided Route Discovery, and in particular on unauthorized modification of QoS parameters.

An attacker can attempt to be chosen as the preferred route by claiming more resources on paths that traverse the attacker. However, with SQoS, an attacker is unable to claim significantly higher resources than the path to the attacker; at most, an attacker can claim the maximum amount of resources that are authenticated with the step authenticator the attacker received. For example, if an attacker receives an authenticator for bandwidth that can represent between 123.65 and 128 kbps, and the actual bandwidth up to the attacker is 125 kbps, the attacker can at most claim slightly less than 128 kbps. As a result, the discovered resource level will be the same as the actual resource level, plus or minus the granularity of a step size, except that any resource limits at the attacker will not be discovered. Cryptographic mechanisms generally cannot force the correct measurement of resources limits at the attacker, since the attacker can intentionally measure resource limits incorrectly.

Multiple colluding attackers can forward metric authentication data from the first attacker on the path to the last attacker on the path, thus allowing the attackers to pretend that there is a direct link with infinite resources between each pair of attackers. In general, an attacker can tunnel packets between nodes controlled by the attacker; SQoS does not defend against this attack.

5.2 SECURING QOS-GUIDED ROUTE DISCOVERY USING PUBLIC KEY CRYPTOGRAPHY

Awerbuch et al [1] proposed a secure routing protocol that uses digital signatures at each hop to ensure that the proper metrics are added. In their scheme, each node signs their address and updates the metrics. Since the metric of each link is known to all nodes, any node can verify the correct path metric, assuming that all nodes on the path are correctly listed. In their protocol, a node forwards any ROUTE REQUEST packet reflecting a better path than the previously forwarded REQUEST packets, which, as described in Section 4.3, can in worst case result in an exponential number of ROUTE REPLY packets in response to a single Route Discovery. We chose to use Awerbuch's scheme in our comparison because it is the only published secure ad hoc network routing protocol that finds paths based on metrics other than hop count and latency, and is thus readily modified to fit the requirements of QoS-Guided Route Discovery.

We modify Awerbuch et al's scheme to create PK-Squared, our public-key based secure QoS-Guided Route Discovery mechanism. In PK-Squared, each node signs the current QoS metric up to and including itself. That is, each node specifies the cumulative metric for the subpath ending at that node. When forwarding a ROUTE REQUEST, the node retains the signatures of all the other hops on the path, and appends its own signature. Since our QoS metrics are monotone, each node must claim a metric not better than the previous node's signature. A REQUEST not satisfying this property can be discarded as malicious.

When a node using PK-Squared receives a ROUTE REQUEST, it checks to see if it has previously heard a REQUEST from this Route Discovery with a better path. If it has, then it silently discards the REQUEST; otherwise, it authenticates each signature on the REQUEST (possibly consulting a cache to speed up the verification of the initiator's signature). It then reduces the QoS parameters in the REQUEST, signs the resulting REQUEST, and forwards it.

5.3 COMPARISON METHODOLOGY

To compare the efficiency of SQoS and PK-Squared, we timed the operation of the primitives of SQoS and PK-Squared on a Mobile Pentium 4 CPU running at 1.6 GHz, a processor commonly used in current laptop computers. Our PK-Squared operations were based on 1024-bit RSA, as implemented in the Crypto++ library. For a hash function, SQoS uses the Rijndael block cipher [11] in the construction standardized by ISO/IEC 10118-2, which was originally proposed by Matyas, Meyer, and Oseas [26]. For our purposes, this construction results in the hash function

Table 1: CPU Costs of SQoS Compared to PK-Squared on a 1.6 GHz Pentium 4

	SQoS	PK-Squared
Initiate Route Discovery	885 μs	7669 μs
Signature Overhead (bytes)	230	128
Accept initiator signature	645 μs	401 μs
Reject invalid signature	34 μs	401 μs
Additional cost to accept ℓ-hop REQUEST	573 μs	$(\ell-1)\cdot$ 401 μs
Packet forward after verification	0 μs	7669 μs
Bytes added at each step	4	132
Number of packets per discovery	$5n$	$2^{n/2+1}$
Hop-Drop Attack Helps Attacker	No	Yes
Metric granularity	300 Steps	Infinite

$H(x) = E_K(x) \oplus x$, where K is a well-known key. For timing this construction, we built our hash function implementation on top of Gladman's implementation of Rijndael [13].

5.4 RESULTS

In SQoS, a node initiating a Route Discovery must sign a REQUEST using its MW-chain. This signature requires 172 hash functions on average, or 322 μs on the 1.6 GHz Pentium 4, and the signature is 230 bytes. We also need to compute hash chains totaling at most 300 steps, which takes up to 563 μs, for a total initialization cost of 885 μs. By comparison, the single RSA signature required by PK-Squared takes 7669 μs, and takes 128 bytes.

Each node needs to verify a Route Discovery only once, after which it can cache the hash of the immutable fields of the REQUEST, which makes verification much faster. To verify the initiator's signature in In SQoS, this verification takes 645 μs on average, whereas in PK-Squared, this verification is a 1024-bit RSA verify, which takes 401 μs. However, rejecting an invalid signature in SQoS is much faster, since SQoS uses 19 separate hash chains, any one of which can be used to discard the signature. As a result, an invalid signature takes only 34 μs (amortized) to reject. In particular, since a correctly verified hash chain element from an invalid signature can be cached, it speeds up the verification of the next valid signature, and represents a cost which would otherwise be a part of the next successful verification.

Once the first REQUEST from a Route Discovery has been verified, SQoS requires only that a receiver verify up to 300 hash chain steps at a cost of up to 563 μs, and compute the evaluation function, which we assume has bounded cost of 10 μs. By contrast, PK-Squared requires the node verify each signature on the path, for a total cost of $(\ell-1)\cdot 401$ μs when the path is of length ℓ.

When actually forwarding a REQUEST, a node using SQoS has already followed all hash chains to their anchors, and thus can use that precomputation to generate the authenticators at no cost. By comparison, when forwarding a REQUEST in PK-Squared, a node must perform an RSA signature, at a cost of 7669 μs. Furthermore, a node forwarding a REQUEST using SQoS increases the packet length by only the length of an address, which in IPv4 is 4 bytes. In PK-Squared, the node must add its address and its signature, which together are 132 bytes, in addition to any QoS metric values which it has changed, which are needed to verify previous signatures. For example, if a node using PK-Squared increases path latency from 20 ms to 30 ms, it must place the new 30 ms time in the packet, but must also include the old 20 ms value, which is needed to verify previous signatures.

One type of attack against ad hoc network routing protocols is the *hop drop attack*, in which a forwarding node removes a previous node from the source route. In SQoS, this hop drop attack is not prevented (although our scheme can be used together with a protocol like Ariadne, which does prevent hop drop), but it does not allow the attacker to claim a better metric than it has. In particular, since the metric authenticators are not tied to the node list, dropping nodes from the node list does not affect what metrics the attacker can claim. By contrast, in PK-Squared, an attacker that drops a hop, particularly a slow hop, can claim significantly better routes than a legitimate node that does not drop a hop.

One benefit that the public key scheme does have, however, is that it is capable of authenticating unlimited metric granularity, whereas our scheme allows only 300 steps. However, we believe that 300 steps provides a good tradeoff between performance and step granularity; in our example, we bounded bandwidth to within 1.5% and latency to within 2 ms. Table 1 summarizes our results. Figure 4 shows the worst-case total CPU time used for a Route Discovery in a network where the topology is the topology described in Section 4.3 and illustrated in Figure 1.

In this evaluation, we have focused on the cost of the mechanism rather than on the performance of a system using such a mechanism. The main reason for not performing this evaluation is that determining the available resources at a node remains an open research challenge. In addition, Route Discovery is relatively rare, and we can impose a limit on the number of Route Discoveries initiated by any node during any time period, which limits the impact of the additional overhead within each ROUTE REQUEST packet. Finally, the additional overhead is quite small. In particular, for each QoS metric, we add a step count, step division (log or linear) and authenticator, which is at most 12 bytes. The postfix evaluation function language can be represented very compactly; if each operation or small literal (under 200) can be represented in one byte, then the evaluation function from Section 4.4 can be represented within 13 bytes.

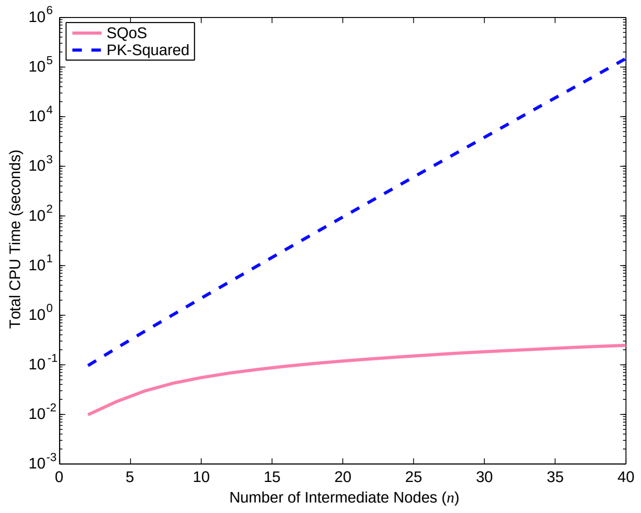

Figure 4: Worst-case CPU usage per Route Discovery in the ad hoc network shown in Figure 1 with *n* total intermediate nodes between Route Discovery initiator and target (logarithmic scale)

6. RELATED WORK

A number of distance-vector routing protocols have been proposed that use hash chains to prevent malicious nodes from reducing the advertised distance to the destination. For example, SEAD proposes the use of a hash chain to secure metric and sequence number [17], and SAODV proposes the use of a hash chain in each packet with the anchor authenticated using a public key signature [39]. In this paper, we extend this approach to metrics other than a hop count. We also introduce a methodology for securely choosing a route other than the minimum-latency route in an on-line fashion. In particular, our approach does not require a node to buffer REQUESTs, since an attacker may attempt to exploit this buffering to overflow the memory of legitimate nodes.

Awerbuch et al [1] propose a security-centric routing which uses digital signatures at each hop to ensure the proper metrics are added. Their scheme enforces the correct metric computation at each hop, since the sender provides these metrics in the ROUTE REQUEST. Our scheme does not require that local metrics be known by other nodes, and does not require the overhead of a digital signature at each hop. Their scheme also does not attempt to prevent an exponential number of packets from being sent in response to a single Route Discovery.

7. CONCLUSION

In this paper we have presented and evaluated SQoS, an efficient secure on-demand QoS-Guided Route Discovery protocol that can be applied to protocols such as DSR [20], AODV [31], Ariadne [18], ARAN [36], and SAODV [39]. SQoS is efficient, relying entirely on *symmetric* cryptography, as symmetric cryptographic primitives are three to four orders of magnitude faster (in computation time) than asymmetric cryptography.

Since most QoS metrics of interest are monotone, SQoS is designed to prevent an attacker from arbitrarily reducing metrics that should be monotonically increasing, and to prevent an attacker from arbitrarily increasing a metric that should be monotonically decreasing. As a result, an attacker cannot claim a metric significantly better than it has heard. Without tamper-proof secure hardware, it is impractical to force each node to claim a correct metric value for arbitrary metrics, but using our technique in SQoS, an attacker cannot gain an arbitrary advantage over non-attacking routes. In addition, SQoS uses a novel, generally applicable technique that combines ROUTE REQUEST authentication and rate-limiting. Finally, SQoS provides the initiator of a Route Discovery with control over which ROUTE REQUEST packets to forward at each node, thus preventing a potentially exponential number of REQUESTs from being forwarded in response to a single Route Discovery as is possible in other protocols; this control increases efficiency and prevents an opportunity for a Denial-of-Service (DoS) attack based on using the routing protocol to easily consume all network resources.

ACKNOWLEDGEMENTS

This research was supported in part by the U.S. Department of Homeland Security (DHS) and the National Science Foundation (NSF) under grant ANI-0335241, by the NSF under grants CCR-0209204, CNS-0325971, and CNS-0338856, and by a gift from Schlumberger. The views and conclusions are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of DHS, NSF, Schlumberger, UC Berkeley, Rice University, or the U.S. Government or any of its agencies.

8. REFERENCES

[1] Baruch Awerbuch, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. An On-Demand Secure Routing Protocol Resilient to Byzantine Failures. In *ACM Workshop on Wireless Security (WiSe)*, September 2002.

[2] John Bellardo and Stefan Savage. 802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions. In *Proceedings of the USENIX Security Symposium*, pages 15–27, August 2003.

[3] Bhargav Bellur and Richard G. Ogier. A Reliable, Efficient Topology Broadcast Protocol for Dynamic Networks. In *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '99)*, pages 178–186, March 1999.

[4] Bob Braden, David Clark, and Scott Shenker. Integrated Services in the Internet Architecture: an Overview. RFC 1633, June 1994.

[5] Josh Broch, David B. Johnson, and David A. Maltz. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Internet-Draft, `draft-ietf-manet-dsr-03.txt`, October 1999. Work in progress. Available from `http://www.monarch.cs.rice.edu/internet-drafts/draft-ietf-manet-dsr-03.txt`.

[6] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the Fourth Annual International Conference on Mobile Computing and Networking (MobiCom 1998)*, pages 85–97, October 1998.

[7] Derya H. Cansever, Arnold M. Michelson, and Allen H. Levesque. Quality of Service Support in Mobile Ad-Hoc IP Networks. In *Proceedings of the Military Communications Conference (MILCOM 1999)*, pages 30–34, October 1999.

[8] Shigang Chen and K. Nahrstedt. Distributed Quality-of-Service Routing in Ad Hoc Networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1488–1505, August 1999.

[9] Don Coppersmith and Markus Jakobsson. Almost Optimal Hash Sequence Traversal. In *Proceedings of the Fourth Conference on Financial Cryptography (FC '02)*, Lecture Notes in Computer Science, 2002.

[10] Douglas S. J. De Couto, Daniel Aguayo, John Bicket, and Robert Morris. A High-Throughput Path Metric for Multi-Hop Wireless Routing. In *Proceedings of the Ninth Annual International Conference on Mobile Computing and Networking (MobiCom 2003)*, pages 134–146, September 2003.

[11] Joan Daemen and Vincent Rijmen. AES Proposal: Rijndael, March 1999.

[12] Shimon Even, Oded Goldreich, and Silvio Micali. On-line/off-line digital signatures. In *Advances in Cryptology — CRYPTO "89*, edited by Gilles Brassard, pages 263–277. Springer-Verlag, 1989. Lecture Notes in Computer Science Volume 435.

[13] Brian Gladman. Cryptography Technology: Implementations of AES (Rijndael) in C/C++ and Assembler, June 2002. Available at `http://fp.gladman.plus.com/cryptography_technology/rijndael/`.

[14] Neil M. Haller. The S/KEY One-Time Password System. In *Proceedings of the 1994 Symposium on Network and Distributed Systems Security (NDSS '94)*, pages 151–157, February 1994.

[15] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Efficient Security Mechanisms for Routing Protocols. In *Proceedings of the 2003 Symposium on Network and Distributed Systems Security (NDSS '03)*, February 2003.

[16] Yih-Chun Hu and David B. Johnson. Implicit Source Routing in On-Demand Ad Hoc Network Routing. In *Proceedings of the Second Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, pages 1–10, October 2001.

[17] Yih-Chun Hu, David B. Johnson, and Adrian Perrig. SEAD: Secure Efficient Distance Vector Routing in Mobile Wireless Ad Hoc Networks. In *Fourth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '02)*, pages 3–13, June 2002.

[18] 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.

[19] Per Johansson, Tony Larsson, Nicklas Hedman, Bartosz Mielczarek, and Mikael Degermark. Scenario-based Performance Analysis of Routing Protocols for Mobile Ad-hoc Networks. In *Proceedings of the Fifth Annual International Conference on Mobile Computing and Networking (MobiCom 1999)*, pages 195–206, August 1999.

[20] 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.

[21] 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.

[22] David B. Johnson, David A. Maltz, and Yih-Chun Hu. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks. Internet-Draft, `draft-ietf-manet-dsr-09.txt`, April 2003. Work in progress.

[23] Leslie Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, November 1981.

[24] Seoung-Bum Lee and Andrew T. Campbell. INSIGNIA: In-Band Signaling Support for QoS in Mobile Ad Hoc Networks. In *Proceedings of the 5th International Workshop on Mobile Multimedia Communications (MoMuC'98)*, October 1998.

[25] David A. Maltz. Resource Management in Multi-hop Ad Hoc Networks. Technical Report CMU-CS-00-150, School of Computer Science, Carnegie Mellon University, 1999.

[26] Stephen Matyas, Carl Meyer, and Jonathan Oseas. Generating Strong One-Way Functions with Cryptographic Algorithm. *IBM Technical Disclosure Bulletin*, 27:5658–5659, 1985.

[27] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology — CRYPTO "87*, edited by Carl Pomerance, pages 369–378, Berlin, 1987. Springer-Verlag. Lecture Notes in Computer Science Volume 293.

[28] Ralph C. Merkle. A certified digital signature. In *Advances in Cryptology — CRYPTO "89*, edited by Gilles Brassard, pages 218–238, Berlin, 1989. Springer-Verlag. Lecture Notes in Computer Science Volume 435.

[29] George C. Necula and Peter Lee. Safe Kernel Extensions Without Run-Time Checking. In *2nd Symposium on Operating Systems Design and Implementation (OSDI '96), October 28–31, 1996. Seattle, WA*, edited by USENIX, pages 229–243, Berkeley, CA, USA, 1996. USENIX.

[30] Kathleen Nichols, Steven Blake, Fred Baker, and David L. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474, December 1998.

[31] 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.

[32] Charles E. Perkins, Elizabeth M. Royer, and Samir R. Das. Quality of Service for Ad hoc On-Demand Distance Vector Routing. Internet-Draft, `draft-ietf-manet-aodvqos-00.txt`, July 2000. Work in progress.

[33] Amir Qayyum, Laurent Viennot, and Anis Laouiti. Multipoint Relaying: An Efficient Technique for flooding in Mobile Wireless Networks. Technical Report Research Report RR-3898, INRIA, February 2000.

[34] Leonid Reyzin and Natan Reyzin. Better than Biba: Short One-Time Signatures with Fast Signing and Verifying. In *Information Security and Privacy — 7th Australasian Conference (ACSIP 2002)*, edited by Jennifer Seberry, number 2384 in Lecture Notes in Computer Science. Springer-Verlag, July 2002.

[35] Pankaj Rohatgi. A Compact and Fast Hybrid Signature Scheme for Multicast Packet Authentication. In *Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS '99)*, pages 93–100, November 1999.

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

[37] David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden. A Survey of Active Network Research. *IEEE Communications Magazine*, 35(1):80–86, January 1997.

[38] Hannan Xiao, W.K.G. Seah, A. Lo, and K.C. Chua. A Flexible Quality of Service Model for Mobile Ad-Hoc Networks. In *Proceedings of the IEEE 51st Vehicular Technology Conference (VTC Spring 2000)*, volume 1, pages 445–449, May 2000.

[39] Manel Guerrero Zapata and N. Asokan. Securing Ad Hoc Routing Protocols. In *Proceedings of the ACM Workshop on Wireless Security (WiSe)*, pages 1–10, September 2002.