MITIGATING DENIAL-OF-SERVICE ATTACKS IN MOBILE AD HOC NETWORKS USING NETWORK CAPABILITIES

by

Eric J. Swankoski
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Computer Science

Committee:

_____	Dr. Sanjeev Setia, Dissertation Director
_____	Dr. Robert Simon, Committee Member
_____	Dr. Angelos Stavrou, Committee Member
_____	Dr. Songqing Chen, Committee Member
_____	Dr. Brian Mark, Department Chair
_____	Dr. Kenneth Ball, Dean, Volgenau School of Engineering

Date: _____ Summer 2017
George Mason University
Fairfax, VA

Mitigating Denial-of-Service Attacks in Mobile Ad Hoc Networks using Network Capabilities

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Eric J. Swankoski
Master of Science
The Pennsylvania State University, 2004
Bachelor of Science
The Pennsylvania State University, 2002

Director: Dr. Sanjeev Setia, Professor
Department of Computer Science

Summer 2017
George Mason University
Fairfax, VA

Dedication

I dedicate this dissertation to my wife, Danielle, who has always pushed me to be my best, and to my daughter, Cora, who has shown me what life is truly about.

Acknowledgments

I would like to thank the following people who made this dissertation possible. First, I extend my gratitude to the U.S. Naval Research Laboratory, my first employer after graduation, whose outstanding continuing education programs allowed me the opportunity to pursue the highest education while contributing to the Navy's mission. Second, and perhaps most importantly, I would like to thank my advisor, Dr. Setia, who always pushed for the highest standards of integrity and academic discovery in our work.

Table of Contents

List of Tables

List of Figures

Abstract

MITIGATING DENIAL-OF-SERVICE ATTACKS IN MOBILE AD HOC NETWORKS USING NETWORK CAPABILITIES

Eric J. Swankoski, PhD

George Mason University, 2017

Dissertation Director: Dr. Sanjeev Setia

The open nature of mobile ad hoc networks (MANETs) makes them vulnerable to denial-of-service attacks. With no well-defined access points, network perimeter, or centralized authority, these networks are susceptible to attacks from one or more authorized nodes (insiders) or malicious external entities (outsiders). Mitigation methods for such attacks are critically important, and in this work we explore the use of network capabilities to enforce a deny-by-default network access control policy. While capabilities can minimize the damage caused by malicious adversaries, the aforementioned characteristics of MANETs also complicate the operation of capabilities. Traditional network capability mechanisms are not designed to cope with frequent route changes. The problem is not well-studied, either for unicast-based or multicast-based MANET communication.

For unicast networks, we have developed EPIC (Efficient Path-Independent Capabilities), a method which combines reverse-disclosure hash chains, identity-based cryptography, and per-packet verification to support the establishment of destination-controlled path-independent capabilities and to show how they can be efficiently operated and maintained in a high mobility environment. EPIC decouples the capability from any particular route, allowing for a seamless transition from one authorized route to another between a source

and destination. We have also developed RAC (Route-Adaptive Capabilities), which uses the same basic building blocks of EPIC but combines the high security of route-dependent capabilities with dynamic route reconfiguration to maintain high efficiency and performance. For multicast networks, we have developed EPIC-M (Multicast), which builds on the core aspects of EPIC to provide capability functionality in multicast networks. EPIC-M uses the building blocks of EPIC, but also utilizes threshold cryptography (partial digital signatures) to facilitate capability establishment and decouple capability establishment and maintenance from multicast routing and membership operations. We show through simulations that EPIC, RAC and EPIC-M can operate efficiently in well-behaved networks while also minimizing network disruption caused by malicious entities in hostile or unsecured networks.

Chapter 1: Introduction

Mobile ad hoc networks (MANETs) offer a high degree of flexibility in many regards, offering substantial advantages over fixed-topology networks. This flexibility, however, leads to additional security complications. The unconstrained physical structure of MANETs does not lend itself well to a secure architecture - with no well-defined access points or network perimeter, any node, malicious or benign, can enter the network and attempt to launch attacks - many of them particularly difficult to mitigate.

In recent years, there has been a great deal of research on securing MANETs [1, 2]. Several authors have proposed novel approaches for secure key establishment, secure neighbor discovery, and secure routing, among others [3–6]. A critically important issue that still remains to be fully addressed is the problem of preventing or mitigating denial-of-service attacks launched by malicious or compromised nodes. DoS attacks in MANETs are both difficult to detect and defend against; nodes may act alone to conduct denial-of-service attacks or they may collude with other malicious nodes for maximum effect.

Attacks can be launched both by outside adversaries (entities external to the network) and insider adversaries (nodes that are members of the target network). Insider attacks are particularly problematic as these nodes are compromised or malicious nodes that possess the proper authorization and credentials to access the network and as such are difficult to differentiate from legitimate participants. Most existing security mechanisms are designed to protect against outsider attacks, but insiders - authorized network nodes - are more difficult to detect and defend against.

1.1 Statement of the Problem

The primary objective of this dissertation is to address the problem of denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks in mobile ad hoc networks (MANETs). While security in mobile ad-hoc networks (MANETs) remains a hot topic in general, the problem of preventing or mitigating DoS attacks in MANETs remains open. Consider, for example, the situation where a node that has been compromised by an adversary injects packets into the MANET with the goal of depleting the resources of the nodes relaying the packets. While researchers have proposed mechanisms for preventing outsider adversaries from launching such attacks in MANETs, these mechanisms do not prevent insider adversaries from launching the same attacks [7, 8]. Given the limited bandwidth and resource constraints of nodes in MANETs, even a single compromised node can launch an effective DoS attack that affects the entire MANET.

As a preventative measure against DoS and DDoS attacks, a *deny-by-default* security policy has been proposed in wired networks [9]. We define this policy as follows: if no specific authorization exists for a given node or traffic flow, then all traffic associated with that node or traffic flow is blocked. This helps to raise the network's resistance to both denial-of-service (DoS) and degradation attacks. Contrast this with the open access policy of the Internet, in which traffic is generally forwarded unless specifically blocked. In this regard, deny-by-default is the opposite of the Internet - all traffic is blocked unless specifically authorized, while in the Internet all traffic is allowed unless specifically blocked.

An important component of the solutions that have been proposed is the use of network capabilities, which are defined as tokens of authorization associated with a network flow that are issued by receivers to authorized senders. These tokens are embedded in each packet that is part of a network flow from the sender to the receiver. Routers on the path from the sender to the receiver drop packets that do not include a legitimate capability, thus ensuring that only authorized traffic can flow between the source and destination.

Most approaches were designed for high-powered modern wired network environments and assumed the presence of trusted central authorities or dedicated routers. Authorization

tokens were embedded in each packet limited to a dedicated route specified at the time of the initial request and authorization.

In this dissertation, we investigate the design of a deny-by-default network access control policy in MANETs based upon the use of **network capabilities**, which represent non-permanent cryptographically secure authorization tokens allowing nodes access to network resources. There are several characteristics of MANETs that make it necessary to explore new approaches for network capabilities different from those proposed for wired networks. First, and most importantly, routes in MANETs change frequently due to node mobility. Most proposals for capability-based protocols depend upon routers on the path between the sender and receiver maintaining state that enables them to verify a capability. Frequent route changes make it necessary for any capability-based protocol for MANETs to re-establish this state as efficiently as possible. Second, nodes in MANETs are not dedicated routers, and are likely to be much more resource-constrained than routers in a wired network. Third, the mechanisms for implementing capabilities are likely to be very different depending on whether the traffic is unicast or multicast.

1.2 Summary of Contributions

Our research, presented in this dissertation, addresses the problem of deny-by-default capability architectures in MANETs. We propose variants of capability mechanisms suitable for both unicast and multicast networks, and we show that capabilities can be used to ensure efficient operation in mobile networks.

1.2.1 Capabilities for Unicast Communication

The first research thrust of this dissertation is in the area of capabilities for unicast-based traffic. Unicast capabilities are point-to-point (single source to single receiver) and can be controlled by the intended receiver. Our contributions are summarized below:

- We introduce two protocols for unicast traffic - EPIC (Efficient Path-Independent

Capabilities) and RAC (Route-Adaptive Capabilities) - to support capabilities in MANETs. EPIC and RAC are independent of the routing protocol. EPIC combines reverse-disclosure hash chains, identity-based cryptography, and per-packet verification to support the establishment of destination-controlled path-independent capabilities and to show how they can be efficiently operated and maintained in a high mobility environment. EPIC completely decouples the capability from any particular route, allowing for multiple simultaneous authorized routes between a source and destination. RAC maximizes the security of the mechanism by creating a hybrid between the fully route-independent method and traditional route-coupled methods, allowing only one active route while supporting dynamic and transparent changes to the authorized route.

- We evaluate the performance of both EPIC and RAC through simulation, showing how they can be used with two different routing protocols - AODV and OLSR - as well as with multiple pedestrian and vehicle-based mobility models. Simulation results show that EPIC provides as much as a 27.3% increase in performance and a 33.7% increase in efficiency over route-dependent capabilities (RDC), while RAC provides higher security and resistance to attack at a cost of slightly reduced performance with respect to EPIC. RAC provides as much as an 8.2% increase in performance and a 13.6% increase in efficiency over RDC. Comparisons to applications that do not use any security mechanism show that EPIC does not incur substantial performance penalties; FTP throughput is reduced on average by 4.2% when EPIC is used and 9.6% when RAC is used.

- We also evaluate the security of the EPIC and RAC mechanisms, from relatively simple attacks (such as capability-enabled local packet flooding) to more complex attacks (such as capability-enabled remote packet injection). Our protocols are independent of the routing protocols and as such are capable of working in conjunction with existing security mechanisms.

1.2.2 Capabilities for Multicast Communication

The second research thrust of this dissertation is on the extension of path-independent capabilities to network environments with predominantly multicast-based traffic. Mobile ad hoc networks are often studied with a primary focus on unicast communications, but anecdotal evidence suggests that communication consists of both unicast and multicast traffic. It is not difficult to envision multiple scenarios where the traffic is predominantly multicast. With one-to-many and many-to-many communication models being essential to a wide variety of MANETs deployed in military or disaster relief applications, providing capability support to multicast communication is necessary for supporting a deny-by-default network communication model. We summarize our contributions below:

- We propose EPIC-M (EPIC-Multicast), a capability protocol that operates independently of the multicast routing protocol. In our approach, group controllers are responsible for issuing capabilities to members of a multicast group. Representing a logical extension of EPIC, it is completely decoupled from the routing protocol. EPIC-M uses threshold cryptography mechanisms to support capability establishment while providing resilience to attack, failure, or compromise of individual group controllers.

- We analyze both tree-based and mesh-based multicast protocols to evaluate the performance and efficiency of EPIC-M. We study three uniquely different protocols: MAODV (tree-based), ODMRP (mesh-based), and PIM-SM, which is tree-based but with the caveat that it can span a very large area. We compare EPIC-M using a point-to-multipoint threshold capability establishment and MCBR traffic patterns against regular multicast traffic without capabilities and show that impacts on performance and efficiency are relatively minimal with respect to multicast networks not employing a capability mechanism, with performance decreasing by an average of 4.0% and efficiency decreasing by an average of 2.0%.

- We analyze the security of the EPIC-M mechanism, considering both simple and complex attacks as well as attacks dependent on the type of multicast routing protocol.

1.3 Structure of the Dissertation

The remainder of this dissertation is structured as follows. Chapter 2 discusses related work on capabilities, including their application to wired networks, mobile ad hoc networks, both unicast and multicast environments, and the secure implementation of capabilities. Chapter 3 outlines the design, implementation, security and simulation of EPIC, our unicast capability mechanism, and its route-dependent adaptive variant RAC. Chapter 4 details the design, implementation, security and simulation of EPIC-M, our multicast capability mechanism. Finally, Chapter 5 outlines directions for future work and provides a conclusion to our work.

Chapter 2: Background and Related Work

Most of the previous research on the use of capabilities has focused on wired networks; there is relatively little work on how they can be used for disallowing unauthorized traffic in MANETs.

Capabilities are defined as a token agreed upon between a sender and receiver that proves the sender is authorized to communicate with the receiver. They serve as a defense mechanism against attacks aimed at preventing legitimate use of a limited resource (such as a communication channel) by flooding it. If an unauthorized malicious user has no capability to send to a particular destination, then it can only flood that destination to the extent that unauthorized traffic is allowed. This prevents attacks by ensuring that the majority of authorized traffic can proceed with little or no interruption even in the presence of an active attacker.

2.1 Capabilities and Denial-of-Service Prevention in the Internet

In an early paper on the subject, Anderson et al defined capabilities as a token agreed upon between a sender and receiver that proves the sender is authorized to communicate with the receiver [10]. They assumed that capabilities would be generated using a hash chain of arbitrary length, with the last value serving as the first capability. As any given capability value was about to expire (based either on a use counter or a time value), the destination would send the next value. Intermediate nodes could then verify this value by calculating its hash and ensuring it matches the previous value. As a necessary step in the establishment of capabilities, nodes on the route between a source and a destination were tied to the capability itself as they were required to verify the capability was authentic.

In wired networks, routes between nodes are relatively stable so associating the capability for a flow with the intermediate nodes on the route between the source and destination does not pose a problem. Intuitively, there is nothing intrinsic to capabilities that would require a specific route. The authors instead propose linking routes and capabilities to both prevent nodes not on the route from interfering and to limit the damage associated with compromised capabilities.

Yaar et al proposed a scheme called SIFF (Stateless Internet Flow Filter) which aimed to prevent or mitigate distributed denial-of-service (DDoS) attacks in the Internet [11]. The method allows receivers to selectively prevent sources from reaching them and also requires that routers be able to differentiate between good traffic and bad traffic, which is accomplished through capability exchange. Also, since each router participates in the identification and verification of a packet, it is in effect a capability system. SIFF made the important contribution of suggesting the linking of a source IP address to a capability hash value, which in this context limited capability abuse to spoofing the source IP address. While SIFF provided some important insights into capability design, as a route-dependent design it still remains inherently unsuitable for mobile ad hoc networks.

Stavrou and Keromytis proposed using stateless multipath overlays to mitigate DDoS attacks [12]. The use of a modified indirection-based overlay network (ION) prevents attackers from discovering connectivity information by utilizing a principle similar to spread-spectrum or frequency-hopping - the source does not utilize the same path on subsequent packets, instead randomly choosing a path from the subset of available paths. While an attacker might be able to attack a specific subset of an overlay network, it is assumed that its attack power is insufficient to bring down the entire overlay network. The basic premise of this approach is that the source has a capability for the overlay network, and packets to the destination are sent on randomly determined paths (making it difficult or impossible for an attacker to prevent delivery). Per-path packet diversity provides some measure of protection against DoS at the cost of increased routing complexity as well as increased latency, but it does offer some resilience to more complex attack models.

Argyraki and Cheriton proposed a scheme called Active Internet Traffic Filtering (AITF), which would allow receivers to identify problematic traffic and request that it be stopped [13]. This requires identification of a particular source's path, which is accomplished through header-contained route records. It also requires that routers be able to provide real-time protection against attackers based on their source. Filtering is based on the interaction between four parties - the source, the destination, and the gateway routers of each. Assuming that the gateway routers remain uncompromised, they then can either request that misbehaving sources stop sending traffic or, if they do not comply, filter the traffic from the malicious entity. Routers may not cooperate for several reasons, both because they may not want to provide diminished service and because they may not have the capability to identify the particular offending flow. Though not called capabilities, the basic principle is the same - provide mechanisms for supporting authorized traffic as well as mechanisms for minimizing the effect of unauthorized traffic. This method has the same drawbacks and weaknesses as the standard capability mechanism - source spoofing, path spoofing, and attacks on the filtering mechanism, for example.

Yang builds on earlier work, including [10, 11, 13], and proposes the Traffic Validation Architecture (TVA), which represents a more thorough definition of a capability-based network architecture [14]. The notion of pre-capabilities is introduced based on earlier work in [11] and requires that participating routers contribute information towards the establishment of a capability. While the end goal remains the prevention of DoS attacks through the control of received traffic by each receiver, the actual implementation and maintenance of capabilities also illustrates a practical defense against DoS attacks. This paper puts forth some key requirements for capabilities, and these requirements apply equally well to both wired and wireless networks - they are granted to the sender by the receiver, they should be both unforgeable and non-permanent, and routers, dedicated or otherwise, must be able to verify them.

The nodes that possess a valid capability are allowed to send a rate-limited amount of data, which helps to prevent attackers with legitimate authorization from disrupting

the network. Following this, unauthorized traffic (packets with an expired capability or no capability at all) is allowed but also at a substantially reduced priority. Such traffic is necessarily required, but is rate-limited generally being allocated a single-digit percentage of the channel. To prevent attacks on the capability setup channel, they propose rate-limiting requests by associating them with a path identifier based on their point of entry into the network. This allows for the timely establishment of new capabilities while providing a reasonable defense against DoS attacks, both against the network at large as well as the capability establishment mechanism. This paper also assumes that capabilities are linked to routes. It also represents a significant advancement in the design of practical protocols that use network capabilities.

Wolf and Vasudevan argue that hop-by-hop capability enforcement with each router participating in both the initial capability setup and the ongoing capability verification does not introduce significant performance limitations after the capability is established [15]. This method advocated the use of Bloom filters to ensure that each router on a path could verify a packet without significantly affecting the data rate. Put simply, the goal is to maintain cryptographic strength (the difficulty of forging a capability) while allowing for quick and efficient verification. Connection setup is difficult and time-consuming, as it requires the sender to sequentially negotiate terms with each router on the path. In static networks, this approach might be acceptable, but is clearly not practical in networks with any significant degree of mobility or resource limitation.

Parno et al first address the issue of changing routes with SNAPP, which allows capability-enabled senders to embed the intended route in a packet's header [16]. This in turn allows established capabilities to continue to function with the established route even after routes are subsequently changed or optimized. While this method avoids capability re-establishment following a route change, it is only viable if the physical route is still operational and as such is only suitable for wired networks.

2.2 Capabilities and Denial-of-Service Prevention in Wireless and Mobile Ad Hoc Networks

The capability protocols proposed for wired networks can be extended to support wireless networks, but there are several key differences that add substantial difficulty. First, routes change frequently due to dynamic node membership and node movement. Second, the wireless channel is significantly less reliable than a wired channel. Third, routers in wireless networks are multipurpose - unlike the Internet, in which routers are dedicated hardware units, MANETs use regular nodes to facilitate routing.

These problems illustrate the necessity for a new approach to capability establishment and maintenance in MANETs. Little work has been done on the subject, with most focusing mainly on the translation of traditional capabilities to what are assumed to be resource-limited mobile nodes. Alicherry et al proposed the extension of existing capabilities to MANETs [17]. This work proposes what are in effect two types of capabilities - globally-issued policy tokens, representing bandwidth allocation, and peer-to-peer capabilities, which represent communication between two nodes. This approach limits the scope of authorized flooding attacks launched by colluding attackers by enforcing global bandwidth policy limitations. Capability-enabled packets are represented by a single static value and are verified probabilistically. This represents a tradeoff between storage requirements (nodes must store policy and bandwidth allocation information for all nodes individually as well as for all communicating pairs of nodes) and computational complexity, but it does serve as an important first step in MANET-focused capabilities. Further work provides some validation for this work, showing through simulation that the approach was workable in MANETs. The effects of MANET mobility are quantified, with a reported throughput decrease of as much as 16.1% and a packet delivery ratio decrease of as much as 9.1% with respect to an open-access network. This work illustrates that capabilities can be adapted to MANETs with acceptable losses to performance. However, this approach does not support a deny-by-default network access policy where any packet that lacks authorization is

dropped.

Later research formalized and improved the approach. Data transmission is assumed to be block-based (multi-packet), with subsequent packet hash signatures transmitted in the initial packet. Subsequent packets consult memory to determine validity of future packets, which represents a tradeoff between computation and storage - no signature verification is required, but state information storage requirements are increased. Results indicate a total performance decrease of between 19.8% and 34%, depending on the situation, while maintaining adequate performance in the presence of attackers utilizing a variety of schemes. However, the DIPLOMA approach has some drawbacks - an offline centralized authority is required to use global capabilities, and the storage requirements are not scalable to larger networks.

2.3 Integrating Denial-of-Service Prevention with Congestion Control Mechanisms

The concept of preventing or mitigating denial-of-service attacks by utilizing existing congestion control mechanisms is an important one, but little work has been done on the subject. In [18], Liu et al introduce NetFence, a method to detect and mitigate DoS attacks by utilizing unforgeable congestion policing feedback structures. Even if a malicious sender has a colluding receiver and attempts to flood the network with traffic, it is unable to forge congestion policing feedback information and as such cannot gain more than its fair share of bandwidth. When congestion is detected and a presumed victim of a DoS attack wishes to stop receiving traffic from a malicious sender, NetFence can act as a sort of capability mechanism and prevent or limit the malicious node from using the network. While NetFence works well within the context of the Internet, many factors make it unsuitable for application in MANETs. For example, it is necessary to identify bottleneck routers. While this is practical in the Internet, node mobility makes it a difficult problem in MANETs. Also, the computational requirements placed on routers to implement NetFence are problematic

without dedicated routers with significant computing power.

Another interesting approach is the combination of traceback and path verification, which aims to prevent deviation from an established or authorized route. The use of traceback allows the point of departure to be identified [19]. One such mechanism, ICING, forwards packets after checking that a particular path is approved and has been followed to that point. Authorization is supported by ID-based cryptosystems, and route deviations result in dropped packets [20]. Other work builds on the concept by reducing the overhead of the mechanism and including the ID of the previous router, allowing for positive identification of a malicious node [21]. The reduction of the problem to binary decisions (to either forward or drop a packet) results in the mechanism in effect achieving the same goal as fully route-dependent capabilities.

2.4 Securing Multicast: Routing and Key Management

While individual networks vary, it has been suggested anecdotally that the majority of communication in MANETs is multicast rather than unicast. Securing multicast communications has focused on multiple different aspects, including authentication (uniquely and securely identifying the members of a multicast group), access control (restricting both membership of the group and the rights of the individual members of the group), and key management (ensuring confidentiality and integrity of group transmissions). Source authentication mechanisms focused on broadcast and multicast, such as TESLA and its variants, change the traditional PKI-based authentication (asymmetric cryptography) in favor of symmetric cryptographic primitives better suited to more resource-constrained networks [22, 23]. Access control and key management can more broadly be combined into multicast group management, which aims to address the problem of limiting access both to group membership and well as to its respective communications. Research on multicast key management mechanisms is fairly broad; approaches can be both high-level (multicast group-level key sharing) as well as low-level (individual node-to-node shared keys).

They can also be fully decentralized or centralized, requiring the use of a key management authority [24–27].

At a higher level, both unique secure multicast mechanisms and additions to existing multicast routing protocols have been proposed. These too are varied, including centralized approaches that are independent of the underlying protocol to tree-based (i.e., MAODV) distributed security mechanisms designed to defend against colluding attackers [28, 29]. Other work focuses on extending specific routing protocols. KHIP, for example, focuses on extending hierarchical multicast routing (HIP) by employing multiple keys per multicast tree [30]. MAODV, in comparison, is relatively well-studied; extensions and modifications have been proposed for source and group authentication [31], secure route discovery [32], and a combined route discovery and maintenance mechanism employing trusted computing mechanisms [33].

2.5 Multicast Capabilities

Network access control for multicast traffic can be divided into two distinct problems - sender control (limiting who can send to a given multicast group) and receiver control (limiting who can receive packets sent to a multicast group) [34]. At a basic level, sender access control is provided by using a source-specific multicast approach, in which receivers can specify from which sources they wish to receive multicast packets [35]. This mechanism is quite similar to that of unicast capabilities, in which receivers explicitly provide transmit permission to sending nodes. Receiver access control can be provided in a variety of ways, but most involve group management via controlling the multicast join process [36]. Very little work has been done on the concept of extending capabilities to multicast. In fact, to date we only know of one paper on the subject, published by Alicherry and Keromytis in 2010 [37]. Like the previous published work, a DIPLOMA-based capability system was implemented in Linux and tested in the Orbit environment. In this work, there are separate capabilities for sending and receiving. Nodes that have a sending capability are also allowed to receive,

while nodes with only a receiving capability cannot send. However, authorized senders can also issue receiving capabilities. Simulation results conducted based on PIM-SM indicate that the multicast DIPLOMA scheme operates with only limited costs in throughput and packet loss (3.6% and 9%, respectively).

Extending capabilities to multicast routing is difficult because unlike unicast routing, network capabilities cannot be practically established between a single source and a receiver. By definition, with many receivers, the problem does not scale and unlike unicast, the peer-to-peer capability establishment approach becomes impractical. This is true of multicast DIPLOMA, which also introduces significant scalability problems - the costs of maintaining and updating capability information, both in terms of storage and communication, increase exponentially as the size of the network increases. It follows that allowing senders to issue capabilities can create security problems - malicious senders with the ability to issue receive capabilities to other nodes could carry out authorized flooding attacks with relative ease. Nonetheless, this paper remains an important first step towards addressing the problem.

2.6 Attacks on Capability Mechanisms

Several solutions have been proposed to address the problem of protecting the initial capability setup. TVA (Traffic Validation Architecture) attempts to secure the capability request mechanism by tagging each request to group and prioritize such requests [14]. Parno et al recognized the need to protect the capability setup channel from attacks. Denying the ability to negotiate a capability in a deny-by-default environment is by definition an effective denial-of-service attack. They propose Portcullis, a method to protect the connection setup phase of a capability system. This is necessary because in a capability-enabled environment, capabilities must be negotiated and established in an unprotected environment. To accomplish this, Portcullis employs a challenge/response model, which bounds the delay any adversary can impose on a legitimate request. Portcullis relies on a trusted third-party entity (in this case, DNS) to periodically release seed values that senders use to generate

puzzles from a known algorithm. Puzzles can be generated at higher difficulties, and priority is given to entities which successfully solve higher-level puzzles. Note that puzzles also include a 64-bit nonce, limiting or preventing the reuse of puzzle solutions. Since an attacker must compute a puzzle solution to access the network, legitimate entities can at the very least access the network (and thus establish a capability) during this computationally bounded time. Results show that the capability establishment mechanism can be protected, allowing access for legitimate participants even in the presence of malicious entities [38].

Argyraki and Cheriton proposed that capabilities were ultimately unnecessary. Their reasoning was that capabilities themselves could be prevented via a denial-of-capability (DoC) attack, and mitigating such an attack would obviate the need for capabilities altogether. This is because some sort of DoS prevention would need to be employed to facilitate legitimate capability establishment. However, this appears to ignore the effects of quality-of-service (QoS) requirements on different aspects of communication. The level of throughput required to sustain capability negotiation is extremely low - in theory, as little as a single packet in both directions. It is also not typically time-sensitive, so extreme levels of packet loss (even asymptotically approaching 100%) do not prohibit capability establishment unless they can prevent delivery of all packets in either direction. As long as packets can eventually be delivered, a capability can be established. Provided that possession of a valid capability can ensure a reasonable quality of service (through higher priority at participating routers), then merely ensuring that capability establishment is possible within some finite and reasonable amount of time would achieve effective prevention against denial-of-capability attacks. Stated simply, preventing DoC attacks requires only that the packet delivery ratio of both the sender and receiver be nonzero. This would not be sufficient for actual application-layer communication, so we operate under the assumption that preventing DoC does not automatically constitute a de facto prevention of DoS. Countermeasures against both DoC and DoS are required in networks and the problems are mutually exclusive [39]. Other potential solutions include LHAP (Lightweight Hop-by-hop Access Protocol), which presents

a mechanism for validating each packet at each node. LHAP is of particular interest because it operates effectively as an additional layer and allows for near-transparent removal of unauthorized or misbehaving nodes from the network [7].

2.7 Decentralized Access Control in MANETs

Threshold cryptography has been studied as a method to produce a single signature or certificate from multiple independent components. As security mechanisms, $\{k, n\}$ threshold cryptosystems allow for a cryptographic primitive to be constructed provided at least k of the n components have contributed partial signatures. Below this threshold k, cryptographic operations are not possible [40]. The applicability of such schemes to MANETs was proposed by Zhou and Haas in [41] to support key management. This extended earlier work by making the scheme more resilient to node failures, accounting for incorrect partial signatures due to compromised nodes and eliminating the requirement for temporal synchronization.

This approach has been since adapted multiple times for use in key distribution and key management mechanisms in MANETs [42, 43]. In these approaches, threshold mechanisms are combined with identity-based cryptosystems with the primary goal of mitigating the single point of failure associated with traditional single-entity key management while also minimizing the storage requirements associated with a traditional public key infrastructure. This makes the approach particularly well-suited to MANETs. Some mechanisms, such as MOCA, aim to provide both availability and security by distributing the key shares to a secure subset of nodes in the network which remain anonymous, thus preventing malicious nodes from attacking those nodes. This provides additional security [44]. Other schemes require all nodes to act as certification authorities, effectively turning access control into a pure peer-to-peer mechanism [45].

In URSA (Ubiquitous and Robust Secure Access Control for Mobile Ad-Hoc Networks), Luo et al put forth an architecture in which all nodes in the network collaborate to certify or revoke each node's network access [46]. A node's local neighborhood is defined as those nodes

within either a one-hop or a two-hop distance, and these neighboring nodes decide based on the node's behavior whether to grant continued access to the network. Misbehaving nodes are unable to access the network once classified as malicious even if they move to a different area of the network. URSA operates independently of the actual misbehavior detection methods and relies on threshold cryptography to establish tickets for network access. This allows any given node to "build" a valid ticket provided enough of its neighbors recognize its valid contributions to the network. The idea of decentralized access control in MANETs is an important concept to our work. While URSA does not specifically deal with network capabilities between a sender and a receiver, it does offer some insights into the ideas of universally verifiable authorization tokens created without a central authority, a key concept in our work. However, URSA is not without problems. Saxena et al illustrated that URSA did not provide for verifiability of the partial signatures used to construct a share. In addition, it showed that RSA-based threshold schemes were inapplicable to MANETs, but other approaches (DSS, Schnorr, and BLS) were both practicable and efficient [47]. We note that each scheme places differing limitations on the number of total threshold nodes as well as the threshold requirement.

Securing MANETs ultimately requires eliminating the dependence of such networks on offline or centralized authorities, and this has led to research in multiple areas on distributed security mechanisms. This complicates the problem, as the limitations of MANET nodes with regards to availability, computing power, and bandwidth make it increasingly difficult to develop distributed security mechanisms without compromising performance.

2.8 Differentiating between our Research and Prior Work

2.8.1 Summarizing Related Work

The main focus of the existing work has been on the application of deny-by-default mechanisms - realized in the form of network capabilities - to wired networks and the Internet. The nature of wired networks allows for computationally demanding and route-dependent

mechanisms, which provide sufficient security given the lack of environmental limitations. However, mobile ad hoc networks exhibit some unique and complicating characteristics that complicate the operation of network capabilities. For example, they have high node turnover, high node mobility, and node heterogeneity, all of which introduce additional complexity. It follows that there is very little work on extending capability mechanisms to MANETs based on their specific characteristics. The DIPLOMA architecture is the main example of a MANET-based capability system, but it does not fully address security concerns in either unicast or multicast networks [17, 37, 48, 49]. We aim to adapt and improve upon some of the existing concepts to create efficient and secure capability mechanisms for both unicast and multicast applications.

2.8.2 Problems with the Existing Framework

It is not difficult to see how the existing capability architecture designed for the Internet would not be suitable for mobile ad hoc networks. One key requirement - that routers must be able to verify capabilities - becomes extremely problematic when we consider that the nodes, which must necessarily also act as routers in MANETs, are neither powerful nor fixed to a given location. As nodes moves and existing routes break, capabilities tied to a specific route would need to be re-established, and thus the risk of failure increases as the route length increases. The problems with MANETs that illustrate the necessity for a new approach to capability establishment and maintenance include high turnover, high mobility, node heterogeneity, a lack of a centralized authority, and mixed multicast and unicast traffic.

Unfortunately, networks with such properties are likely to be deployed in critical situations (such as military or emergency response applications). This makes the need for robust traffic and channel access control even greater. First, and perhaps most importantly, all of these factors lead to high variability among routes over time. Any method that relies on route dependence is necessarily at a disadvantage as it becomes increasingly inefficient as mobility increases. This is the main motivation for addressing unicast capabilities with

a path-independent approach. Second, with a potentially significant amount of multicast traffic in any given MANET, the established architecture fails to fully address the problem as it focuses only on unicast traffic.

2.8.3 Building on the Existing Framework

We can build on the existing research by adapting and modifying prior concepts as well as by developing novel approaches to solving the problem of DoS and DDoS mitigation in MANETs. The end goal is to support the efficient use of capabilities as both a mechanism for node authorization and access control as well as a mechanism for DoS prevention in such networks through per-packet authorization. We propose the use of the following concepts to provide for efficient, high-performance, and high-security capability mechanisms.

1. *Fully Distributed Authorization Infrastructure:* Deny-by-default protocols in MANETs should require no centralized infrastructure to support capability-enabled networks, allowing for both dynamic re-establishment and dynamic global policy changes.

2. *Efficient, Secure, and Available Cryptographic Mechanisms:* The computational and communication efficiency as well as the availability of cryptographic mechanisms can be significantly improved by using identity-based cryptography, which both mitigates the need for public key management systems and provides for universal verification, and threshold cryptography, which mitigates failure points and increase the availability of certification authorities.

3. *Route Independence:* Due to the inherent mobility and unreliable communication patterns of MANETs, deny-by-default mechanisms for both unicast and multicast traffic should either be independent of the route between communicating nodes or tolerant of route changes.

4. *Efficiency and Scalability:* Due to the limited computational and storage capabilities of MANET nodes, mechanisms should be scalable and efficient, requiring minimal

computation to verify a capability and minimal state storage to reference existing capabilities.

5. *Resistance to Misuse and Attack:* Capabilities should be unforgeable, non-permanent, and be independently secure in that they do not provide an opportunity for either insider or outsider attackers to misuse the system.

Chapter 3: Supporting Deny-by-Default for Unicast Traffic in MANETs

In this chapter, we propose two methods to implement a deny-by-default network access policy for unicast capabilities in MANETs: EPIC, a method that combines reverse-disclosure hash chains, identity-based cryptography, and hop-by-hop verification to support the establishment of path-independent capabilities, and RAC, an adaptive extension to EPIC that combines route dependence with seamless dynamic capability maintenance and renewal. For comparison, we study a traditional route-dependent capability framework in which only the originally established route is allowed. We analyze the efficiency, security, and performance of all three protocols in MANETs for a variety of scenarios involving different applications and varying levels of node mobility. We also compare the performance of our protocols against applications that do not employ any security protocols.

3.1 Capability Protocols in Wired Networks

Before we introduce EPIC, it is helpful to consider the operation of capability-based protocols in wired networks. The main steps are described below:

1. *Capability Requests:* Once a node determines it needs to communicate with another node, it checks to see whether it has a capability for that node. If it does not, it issues a request. This request identifies the sender, the terms requested for the capability, and, optionally, the sender's authentication information. Some mechanisms also implement some protection against denial-of-capability attacks such that capability establishment is allowed in a timely manner [10, 11, 14, 15]. Most protocols propose piggybacking capability requests on TCP SYN packets or specialized RTS packets.

2. *Capability Response:* Typically, the destination issues capabilities in response to a request if it is willing to receive packets from the source node [10,11,14]. Alternatives exist where routers rather than destinations issue capabilities provided the sender can successfully answer a challenge [15].

3. *Capability Verification:* Once a sender receives a capability from the source, the capability is included in every packet it sends. Verification of capabilities in wired networks is done by the intermediate routers. It is important to note that in most of the prominent proposals for capability-based protocols in wired networks, the capability itself is closely tied to the route between the source and destination. In [14], routers tag requests with their own identifying information and as such are explicitly identified in the capability itself. In [15], the routers themselves establish the capabilities and use Bloom filters to verify them. In [11], routers verify capabilities by calculating a hash based on a combination of the router addresses as well as the source and destination addresses.

4. *Capability Maintenance and Renewal:* As a general rule, capabilities should be valid only for a limited and defined amount of time. Thus, if a capability is compromised, the damage is limited because even without detection its validity period is finite. When a capability reaches or approaches its expiration, it is renewed either by the destination or one of the routers on the path. Routers can verify that the new capability value and the previous value are assigned to the same communication flow.

3.2 EPIC: Motivating Path Independence

The traditional deny-by-default capability protocols were originally proposed for wired networks, and the majority were dependent on the route between a source and destination for verification - that is, if the route changed, the capability would need to be re-established. This approach is sufficient for traditional wired networks in which the topology changes infrequently. The increase in overhead associated with the generation and verification of

multiple digital signatures did not present a problem because the dedicated routers processing the authorized packets are relatively powerful.

However, route changes in MANETs are more frequent for multiple reasons, including high node mobility, node heterogeneity, and changes in the network membership due to nodes joining or leaving the network. These factors introduce additional complexity as the route between two nodes may change frequently. This is problematic since an existing capability associated with a flow between two nodes becomes invalid once there is a route change, and continued communication necessitates a repeat of the capability establishment process. The limited computational capability also serves as motivation to minimize digital signature generation and verification.

We illustrate this with an example in Figure 3.1. Suppose that A initiates a request for a capability with F and establishes a capability along the path $A \rightarrow B \rightarrow C \rightarrow E \rightarrow F$. Recall that with traditional mechanisms, no consideration is given to the possibility of that route breaking and when routes change, capabilities are simply reestablished along the new route. Now suppose that the route breaks as C leaves and is no longer within transmission range of B. B then repairs the route locally and establishes a route through D. However, when D receives packets containing a capability for the flow $A \rightarrow B \rightarrow C \rightarrow E \rightarrow F$, it drops them as it cannot verify them. The capability is not re-established until A detects, either through explicit notification or lack of acknowledgment, that its route to F needs repair. Node A must then re-initiate the capability negotiation process and it establishes a new capability along the path $A \rightarrow B \rightarrow D \rightarrow E \rightarrow F$. Each time a route changes, this process repeats.

This is precisely the situation we want to avoid. In the ideal case, B detects that C is no longer reachable, repairs the route locally through D, and transmits a packet that includes a capability to F along the new route without requiring any additional information or retransmission from A. This imposes two requirements. First, the capability must not include anything that is tied to C - the capability should be path-independent. Second,

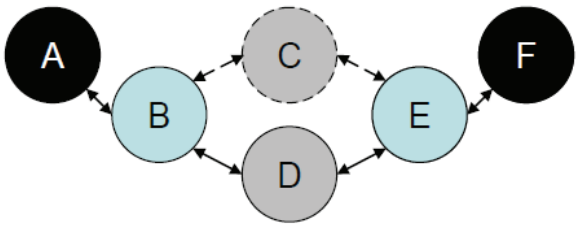

Figure 3.1: Illustrating Capability Maintenance Problems Associated with Node Movement

the information included in the path-independent capability should be unforgeable (cryptographically secure). Route-dependent mechanisms do not meet the first requirement, and this is a major motivation for the development of EPIC. Using route-dependent capabilities would present significant problems as source nodes would be forced to renegotiate capabilities every time a route changes. In EPIC, capabilities by default do not include any route-dependent information. Instead, each capability includes the sender, the receiver, and a hash value, which is part of a one-way hash chain.

3.3 EPIC: An Overview

We propose EPIC as a lightweight path-independent capability protocol, which aims to allow for change in the established route of the capability while simultaneously limiting the ability of malicious nodes to hijack or otherwise misuse legitimate capabilities. Following this, we must support two important concepts - first, that full path independence does not significantly impact performance in the absence of malicious nodes, and second, that full path independence successfully mitigates DoS and DDoS attacks as well as capability hijacking or misuse.

Capabilities are granted by the receiver to the sender following an initial request. Both the request and the response are digitally signed, which ensures that capabilities are unforgeable and authentic. End-to-end encryption ensures confidentiality. The sender is then able to use the capability response to mint its own capabilities on a per-packet basis by

creating a one-way hash chain. Both the initial and terminal values of the hash chain are specified by the receiver in the initial response, providing for full receiver control over the number of packets a sender is authorized for. Each capability-enabled packet includes a capability value that is a predecessor in the one-way hash chain of the previously used value, which link together packets in the flow. This ensures the security of the mechanism provided the hash function is secure as the sender is the only node with access to the starting value in the chain.

The first capability-enabled packet along a new or updated flow includes a digitally signed field known as the base capability, which allows for full authentication of the capability by any member node of the network. When a node receives the first packet that is part of a flow, it verifies the signature for the initial capability to check its validity and stores the base capability. In EPIC, we propose to use digital signatures based on identity-based cryptosystems because of the performance benefits of using identity-based cryptosystems over traditional public key cryptosystems in MANETs [50,51]. Specifically, a node receiving a packet that includes a signed capability can derive the public key of the signing node from its identifier and use that public key to verify the signature. In EPIC, a node needs to verify a signature only when it receives the capability for the first time; for subsequent packets, only a computationally inexpensive hash operation is needed. When a route changes, the signed base capability is forwarded to the new node along with the first packet it receives that is part of the flow. With local capability storage and no route information included in the base capability, EPIC achieves full route independence without requiring any additional communication by either the endpoints or intermediate nodes.

3.4 EPIC: Path-Independent Capability Protocol Design

3.4.1 EPIC: Protocol Design

In this section, we describe the operation of the EPIC capability protocol. Specifically, we discuss the steps followed by the sender, the receiver, and individual nodes along the route

to generate and verify capability-enabled packets.

EPIC Notation

We use the following notation to describe our protocols:

R, S	R and S are communicating principals: S, the source, wishes to communicate with the destination R
N	An intermediate node of the route
ID_X	The identifier for a given node X
K_X^{-1}	Node X's private key
K_X	Node X's public key, derivable from X's identifier ID_X
$\langle M \rangle_{K_X^{-1}}$	Field or message M signed with node X's private key
$E\{M\}_{K_X}$	Field or message M encrypted with node X's public key
f_h	One-way hash function used for generating a hash chain
BC	The base capability
CAP_i	Capability with sequence number i
T_0	Initial capability request time
T_{CV}	Validity period for a given capability chain
$RT_{S,n}$	The route membership between a source S and a given node n
RTT_X	The maximal approximate expected configuration-time round-trip delay based on network area, node quantity, and bandwidth capabilities
T_{GP}	Grace period for forwarding packets without re-established capabilities and establishing new capabilities prior to expiration based on $2 * RTT_X$

EPIC Capability Request

- *S: Compute* $M_1 = (RFC, T_0)$

- $S \rightarrow R$: $\langle M_1 \rangle_{K_S^{-1}}$

To initiate communication with the destination, the source sends a digitally signed and timestamped request for capability (RFC) packet. The capability request packet can be an independent packet or piggybacked on either a routing control packet or a transport protocol packet. There are two security concerns with regards to initial capability requests. First, successful denial-of-capability attacks are effectively denial-of-service attacks and thus we need to protect capability setup. Fortunately, we can adapt methods similar to those proposed for wired networks [10, 11, 14, 15]. Second, if requests do not carry digital signatures, then any node can request a capability for any other node. Initial requests thus require authentication and must be digitally signed. Intermediate nodes forward capability requests to the destination, but they do not need to verify the signature.

EPIC Capability Response

- R: *Compute* CH_n *and* CH_0
- R: *Compute* $BC = (T_0, T_{CV}, ID_S, ID_R, CH_0)$
- $R \rightarrow S$: $E\{\langle BC \rangle_{K_R^{-1}}, \langle CH_n \rangle_{K_R^{-1}}\}_{K_S}$

When a node R receives an authenticated capability request and is willing to accept packets from the source node S, it will calculate the base capability BC and securely transmit it to the source along with additional information as described below. EPIC is based upon the use of reverse-disclosure one-way hash chains. The receiver calculates a value CH_n based on a one-way hash function f_h, which takes as input a unique combination of the sender and receiver identification ID_S and ID_R, the receiver's public key K_R, the initial timestamp T_0, and a nonce RN_0 as shown below. Subsequently, CH_0 is calculated using n successive applications of a one-way hash function f_h to produce a sequence of values $(CH_n, CH_{n-1}, ..., CH_1, CH_0)$.

$$CH_n = f_h(ID_S, ID_R, K_R, T_0, RN_0) \tag{3.1}$$

$$CH_0 = f_h(f_h(...(f_h(CH_n)))) \tag{3.2}$$

The base capability BC is composed of the parameters associated with the network flow for which the capability was requested - namely the node identifiers ID_R and ID_S, the initial timestamp T_0, the validity time period for the capability chain T_{CV}, and the anchor of the one-way hash chain CH_0. Further, BC is digitally signed by the receiver R, so that any node that receives BC will be able to verify its authenticity using the public key of R. We assume the use of identity-based cryptosystems so the public key of R can be derived from its identifier. Node R then transmits its response to S, which includes both BC and CH_n. The base capability BC and the capability root value CH_n are both digitally signed by R and the message encrypted with the sender's public key K_S. This ensures that the sender is the only entity with access to the capability.

EPIC Protocol Operation: Authorized Packets

- *S: Compute Capability Hash Values (CH_n, CH_{n-1}, ..., CH_0)*
- *S: Compute Initial Capability $CAP_0 = \{BC, 0\}$*
- *S: Compute Individual Capabilities $CAP_i = \{CH_i, i\}$*
- *$S \rightarrow R$: Data Packet $\mid CAP_i$*
- ***Intermediate Nodes with no prior knowledge of BC:*** *The sender includes the signed capability BC with the first packet of the flow ($i = 0$). Otherwise, the previous node on the route, having detected a route change (a different next hop), will include the signed BC with the data packet. Verify authenticity of BC with K_R and check that it has not expired. Verify that $CH_0 = f_h^i(CH_i)$ using CH_0 included in BC. Cache BC and CH_i along with the next hop.*
- ***Intermediate Nodes with cached BC:*** *Verify $CH_{i-1} = f_h(CH_i)$. If packet is outdated or a duplicate, drop the packet. Check that capability has not expired using T_0 and T_{CV}. If the next hop has changed, include cached BC in the data packet.*

After receiving the base capability BC and CH_n, node S can calculate its own per-packet capabilities. First, it uses the value CH_n to mint exactly n capability hash values. Once the sender S has these n values, it can construct its per-packet capabilities for the next n packets, associating each successive capability with one packet. Each capability is represented by the value CAP_i and consists of the base capability BC, the current hash value CH_i, and the sequence number i. The basic idea behind this approach is that the hash values CH_i are disclosed in reverse order of their generation. A node that obtains CH_i will not be able to derive CH_{i+1} because of the one-way aspect of the hash function, but will be able to verify that CH_i is part of a hash chain if it has cached a previously disclosed hash value CH_k, where $0 < k < i$.

Normal operation of EPIC does not require any knowledge of the underlying routing protocol. The process for verifying a capability is identical regardless of whether a node is receiving the first packet in an authorized flow or it is receiving a packet from an established flow as a router in a new route resulting from a route repair by the underlying routing protocol. Intermediate nodes upon first receipt of a packet belonging to a flow must then verify the included capability by verifying the authenticity of the base capability BC using R's public key and, if necessary, performing some number of hash calculations to ensure that the included hash value CH_i is part of the same hash chain terminating with the value CH_0 included in BC. If the node has already cached the value of BC, for subsequent packets it needs to verify that the current packet capability hash value CH_i is part of the same chain as the signed capability CH_0. The sequence number i can be used along with the time values T_0 and T_{CV} to determine the approximate number of hash computations required to ensure that CH_i is a predecessor of CH_0 in the hash chain. Finally, it need only perform the verification step.

EPIC Protocol Maintenance: Normal Operation

The only maintenance required for EPIC capabilities is the periodic renewal as one capability approaches the end of its lifespan. The protocol interaction for maintenance between

senders and receivers is identical to that of the initial negotiation. The destination should not proactively send new capabilities as current ones expire. Instead, senders explicitly request renewed capabilities. Requests for renewed capabilities take place as the current capability is expiring. Nodes can request renewed capabilities provided their current capability expires less than or equal to a value of T_{GP} from the present time. The use of T_{GP} along with single-use capabilities helps prevents nodes from stockpiling capabilities and ensures non-permanence.

3.4.2 EPIC Capability Components

Figure 3.2 provides an illustration of what is included in an EPIC capability.

BC: Base Capability	
ID_S	Sender ID
ID_R	Receiver ID
CH_0	Initial Capability Hash Value
T_0	Initial Capability Timestamp
T_{CV}	Capability Chain Validity Period
Receiver Signature S_R	
CH_i	Current Capability Hash Value

Figure 3.2: Components of an EPIC Capability. The encapsulated portion represents the base capability BC, and the receiver signature S_R is computed over this portion. The current capability hash value CH_i is used for subsequent packets in a flow.

- *Receiver Signature - S_R:* EPIC capabilities are signed by the receiver. The receiver's identification information (and thus the means to derive its public key) is known from the packet's routing information. The signed portion - ID_S, ID_R, CH_0, T_0, and T_{CV} - is only verified when first received. The inclusion of the initial timestamp prevents malicious nodes from replaying the capability at a later time. If a node has already seen a capability in this chain, then the node can simply perform one or more hash computations to ensure that the received value is part of the same capability chain.

- *Initial Capability Hash Value - CH_0:* This is considered the initial capability hash values and represents the final element of a one-way hash chain. This value is critical in both ensuring non-permanence of the capability as well as facilitating universal verification. CH_0 is calculated using n successive applications of a hash function f_H to produce a sequence of values $(CH_n, CH_{n-1}, CH_{n-2}, ..., CH_1, CH_0)$, and T_0 is the current time and R_0 is a nonce. The concatenation of these values represents a unique input to the hash function not reproducible by other nodes and also ensures that successive capabilities between a sender/receiver pair will not be recycled.

  A key part of the capability negotiation process is the encrypted transmission of the value CH_n to the sender. The destination securely transmits both CH_n and CH_0 to the sender, which allows the sender to mint exactly n capabilities. If each capability is to be forwarded exactly once, then the sender has authorization to send no more than n packets without requesting additional authorization. By transmitting CH_n, the destination is able to enforce a unique per-packet authorization scheme for exactly n packets without being required to transmit n unique capabilities. This does not compromise our non-permanence and universal verification requirements as the destination must periodically issue new capability values when current values expire. The first value intended for use by the destination is CH_0, and each capability is only valid as long as is indicated by the capability chain validity period T_{CV}. The verification of a capability starts with the verification of the destination's signature using its public key. The current capability CH_t is then verified along with the current time and the capability chain validity period T_{CV}. A node can apply successive applications of the universally known hash function f_h to ensure that the current capability CH_t is a predecessor of CH_0 in the hash chain.

- *Current Time and Capability Values: $CH_i/T_0/T_{CV}$:* By definition, CH_i is the current capability and must be a predecessor of CH_0 in the one-way capability hash chain. T_{CV} represents the length of time any particular capability value in a given capability chain can be considered valid and T_0 represents the initial timestamp of the capability.

With unique per-packet authorization (a single capability hash value per packet), a capability value will only be valid until it is used once or its validity period expires, whichever comes first. In a static route, an intermediate node will have to perform no more than one hash computation at a time to verify that the current value CH_i is the immediate predecessor of the prior known value CH_{i-1} in the chain. Loose time synchronization is required among nodes, and we make the assumption that all nodes in the network are temporally synchronized within some error margin δ such that if the current absolute time is considered to be T_a, then any node in the network will have its own time T_X such that $(T_a - \delta) \leq T_X \leq (T_a + \delta)$. In this manner, at most one additional hash calculation is required in the event the current time T_0 falls within 2δ of the beginning or end of a capability validity period. This is because it is possible for two loosely synchronized nodes to assume they are operating legitimately within two adjacent capability validity periods. The use of single-use capability elements allows for the validity period to be very short, thus providing further protection against abuse. We note that the capability lifetime, however, should be sufficiently long enough that any capability issued to any node will be valid for most of its nominal life; in absolute terms this means that for a given capability lifetime T_{CV} and a time synchronization error margin δ, the relation $T_{CV} \gg \delta$ must hold true.

3.5 EPIC: Security Analysis

In this section, we evaluate the security of the EPIC protocol with respect to various attacks. We use the defined notation from Section 3.4, while the protocol is evaluated under the following assumptions:

1. The digital signature scheme and its underlying identity-based cryptosystem are secure.

2. The one-way hash function used in capability generation and verification is secure.

3. A node's private key is secure, preventing one node from masquerading as another.

4. Senders and receivers are not compromised at the time of capability establishment.

3.5.1 Attacks on Deny-By-Default Protocols

Forgery

In EPIC, capability requests are signed by the originating node S, and thus no node can request a capability on behalf of another node. A receiver R will not generate a capability response without a signed request, preventing unauthorized or outsider nodes without a valid private key from requesting capabilities. The response generated by the receiver R is both digitally signed (ensuring it was generated by the named receiver R) and encrypted (ensuring it can only be used by the named sender S). This signed response, BC, contains the hash chain anchor CH_0. This in turn allows all nodes in the network to verify the authenticity of BC as legitimately generated by R. These signature scheme ensures the generation of capabilities is limited to legitimate senders and receivers.

Interception

The use of end-to-end encryption prevents interception by malicious nodes as the receiver R encrypts the response, including both the signed base capability BC and the signed capability hash chain root CH_n, with the public key of the sender S. Without compromising the sender, no node can intercept a capability intended for another node. Further, as the signed capability BC is tied to a single source/destination pair, it cannot be used for communication between any other pair of nodes. The signed and encrypted value CH_n ensures the sender is the only node with the ability to create individual capability values.

Replay Attacks

Initial capability requests are timestamped by the sender S with the value T_0. Since this request is signed by the originating node S, it cannot be successfully modified by another node. The receiver R generates a signed response that includes the chain validity period

T_{CV}. This is controlled by R, preventing senders from requesting arbitrarily long or indefinite capabilities. During the chain validity period T_{CV}, capabilities are only valid for communication between S and R. Beyond that time, values in the capability chain are invalid, and they cannot be used for communication by any node regardless of identity.

Data Modification

By utilizing a reverse-disclosure hash chain, EPIC allows each capability to be used exactly once. If a malicious node were able to successfully modify a capability-enabled message to add spurious data and send it to the destination (or a node prior to the destination also used by the legitimate sender), it could prevent the legitimate packets from being accepted if the malicious packet reaches the node prior to the sender's legitimate packet or the sender's legitimate packet is dropped or lost. Spurious data packets reaching a node after the use of a legitimate capability will be dropped as duplicates. Since any message can be modified by an intermediate forwarding node, end-to-end message authentication is required, allowing the destination to detect such modification attacks. We note that end-to-end message authentication is required with or without EPIC, so no additional complexity is introduced.

Capability Revocation

Capabilities in EPIC are non-permanent, so expiration (and thus implicit non-real-time revocation) is guaranteed for any given capability. However, should explicit revocation be necessary, EPIC could be modified to allow receivers to maintain final control over the capabilities they issue by allowing for explicit termination. This can be accomplished by broadcasting authenticated explicit revocation messages; nodes can both invalidate existing (cached) capabilities and retain the revocation information for future use.

3.5.2 Flooding and Remote Packet Injection

Traditional route-dependent capability methods have the inherent protection of complete route dependence; packets being routed on unauthorized routes are blocked immediately by intermediate nodes. However, in EPIC, packets injected remotely could be forwarded and accepted by legitimate nodes. A consequence of complete path independence is that the capability mechanism cannot differentiate between normal mobility-induced changes to the route and deliberate misuse. Since capabilities are universally verifiable, any node that receives a capability-enabled packet will attempt to verify the capability and forward the packet. The broadcast nature of the medium allows nodes not on the intended route to both hear and understand these packets just as it allows malicious nodes to forward these packets to their neighbors, causing unnecessary computation for the receiving nodes as well as unnecessary delays for authorized traffic. While EPIC is resistant to capability interception, it is not inherently resistant to authorized flooding attacks in the form of remote packet injection.

However, if colluding attackers are able to utilize a wormhole to inject packets at a remote location, they may use strategic locations to conduct an effective DoS attack against a legitimate sender by retransmitting an overheard or intercepted capability. In Figure 3.3, the sender S communicates with the receiver R. Locally, attackers X and Y are present and inject duplicate capability-enabled packets to nodes C and D. Nodes on the actual route are more likely to reject packets as duplicates - in this case, B can reject duplicate packets. This is because it is more difficult to find a disjoint route locally. However, if X can use a wormhole or other side channel to transmit its packets to Z, a remote colluding attacker, then Z can inject packets to its neighbors E and F (who would ordinarily never see this capability). This allows attackers to exdhaust resources by causing significant congestion regardless of whether the injected packets ever reach the destination or not. The ideal solution would be to prevent disjoint routes from being used by either preventing multipath communication or only authorizing a single active route between each pair of communicating nodes, which represents a tradeoff between increased complexity (identifying

and maintaining authorized route information) and security (mitigating multiple types of capability-enabled packet injection).

If attackers are able to effectively create and exploit wormholes, it is likely they can inject duplicate packets without detection. While wormhole attacks are possible, they are difficult to engineer and represent a fairly complex and impractical attack, and integrating EPIC with secure routing protocols would make detection and mitigation of wormhole attacks more efficient. However, wormhole attacks represent an attack against the routing protocol and not necessarily against EPIC, and secured routing protocols do not have this vulnerability. Regardless, mechanisms such as temporal or geographical packet leashes can be used to detect and mitigate such attacks [52].

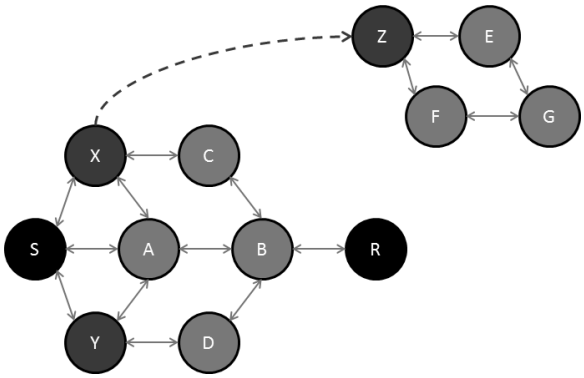

Figure 3.3: Flooding Attacks: Local and Distant Packet Injection

3.5.3 Attacks on Capability Establishment

In ad hoc networks that employ a deny-by-default access policy, preventing legitimate nodes from obtaining authorization effectively creates an absolute denial-of-service attack. We consider neither such denial-of-capability attacks nor their countermeasures in this paper, instead noting that existing measures (such as Portcullis) are adaptable to MANETs [38]. Preventing against denial-of-capability attacks is critical to the successful operation of deny-by-default mechanisms, but it does not need to be a part of such a mechanism. Intuitively,

DoC prevention is achieved provided all nodes that are otherwise connected in the network topology are able to communicate bidirectionally, and we posit that the solution to this problem is best suited for a lower layer of the network. As such, it is beyond the scope of this work.

Certain attacks against the routing protocol, such as selective forwarding, may prevent EPIC from establishing or renewing capabilities. Intermediate nodes, when compromised, might maliciously drop packets that are part of a capability request or response chain. While we note that all capability mechanisms are susceptible to such attacks, certain methods can be used to detect and mitigate malicious packet forwarding activity. In wireless networks, it is difficult to differentiate between legitimate failures (due to collisions and transmission errors) and deliberate failures (caused by malicious nodes). Hayajneh et al show that passive observation can significantly reduce such attacks by identifying forwarding behavior that deviates significantly from expected values. While ultimately dependent on the accuracy with which nodes can identify the expected behavior, forwarding rates that deviate by more than 4% can be detected [53]. Active probing techniques can also be used to monitor the forwarding behavior of a given node. With this method, one particular one-hop neighbor of an evaluated node forces traffic through the target to a different one-hop neighbor. While this is more limited as support for multi-hop source routing is required, it can also effectively identify malicious packet dropping with as much as a 94% accuracy rate [54].

3.5.4 Congestion Control

The establishment and enforcement of capabilities does not prevent all malicious activity by an inside attacker. Multiple colluding nodes could create multiple capabilities, exploit multiple paths, or otherwise attempt to subvert not the capability protocol itself but the limits it places on nodes. Capabilities do not provide a complete defense against collusion. However, queue management and congestion control mechanisms are useful in enforcing a maximal level of fair use, and this applies regardless of whether a deny-by-default mechanism is utilized or not. Thus, the solution to this problem (the general enforcement of

equitable network resource allocation) is orthogonal to the use of capability mechanisms, but it remains a necessary component of a secure network. As such, it is also beyond the scope of this work.

3.6 RAC: Enhancing EPIC Security

EPIC, at its core, is designed to allow for effectively unlimited changes to a route without requiring capability renegotiation. However, as discussed in Section 3.5, when malicious nodes are present, we consider the possibility that capabilities could be hijacked with the goal of disrupting normal network operation. Authorized flooding attacks can present a problem - as more malicious traffic is forwarded with valid capabilities, legitimate communication must contend for channel resources and throughput decreases. To mitigate this problem, we aim to enhance EPIC with security features that limit or prevent the detrimental effects of malicious nodes forwarding otherwise authorized capability-enabled packets elsewhere in the network. In the case where the underlying network is secure against wormholes and similar attacks, EPIC itself is adequate and does not introduce any security vulnerabilities. However, we propose a modification of EPIC with the assumption that a secure underlying network is not guaranteed.

Route-dependent capabilities provide excellent security against such attacks, but at a price - decreased performance and increased overhead. Intuitively, some combination of the two approaches would be ideal. We propose augmenting EPIC with a security mechanism, RAC (Route-Adaptive Capabilities), which employs dynamically repaired path-dependent capabilities. We propose a modification to EPIC - capabilities are limited to only one active route between a sender and receiver, but as the route changes, the capability is dynamically reconfigured to support the new route. Intermediate nodes on the route responsible for forwarding packets, when they detect a route has changed, issue signed capability requests to the receiver R on behalf of the sender S.

3.6.1 Route-Adaptive Capabilities: An Overview

RAC provides for a capability protocol that can adapt to changes in the route with little or no disruption to the operation of capability-enabled packets. This is accomplished by combining the efficient verification aspects of EPIC with the security restrictions of full route dependence. We propose RAC as an extension of EPIC that replaces complete path independence with dynamic route dependence. In RAC, capabilities are associated with the route beween the sender and receiver. During the capability request process, intermediate nodes cache some state information about the process, including the time of request, the sender, the receiver, and the partial route. When routes break, intermediate nodes issue signed requests on behalf of the sender using cached route information. The receiver then issues a capability to the sender using the new route, but with the original expiration time. A grace period T_{GP} allows for packets to be forwarded during the capability re-establishment process. The request and renewal process of the RAC protocol minimizes disruptions to communications even when routes break and also prevents intermediate nodes from hijacking the renewal process by ensuring the non-permanence of both initial and repaired capabilities.

When the source receives and uses the repaired capability, the previous capability is invalidated. By default, RAC only allows for a single flow to be authorized between a sender and a receiver. When a sender receives a capability that it did not request, it must follow that a route change has occurred and one or more new nodes not on the original route have requested a renewal with their own information. If the sender uses this new capability, intermediate nodes will replace their existing cached information with the new capability, and subsequently they will only accept and forward packets with the most recently used capability.

Figures 3.4, 3.5, and 3.6 illustrate graphically the normal operation of the RAC protocol.

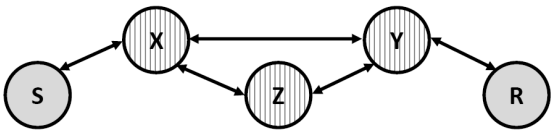

Figure 3.4: The sender S sends a capability request to the receiver R by first creating and sending a signed request packet and forwarding to its next hop X. Before forwarding the request, X caches the current route $RT_{S,X} = \{S, X\}$, the request time, the source and destination S and R, and the next hop Y. It also sets a flag indicating it has forwarded a signed request. Node Y does the same, caching its route as $RT_{S,Y} = \{S, X, Y\}$. The receiver R, after receipt of the capability with route $RT_{S,R} = \{S, X, Y, R\}$, issues a capability to S along this route with an expiration time of T_1.

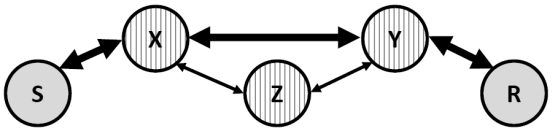

Figure 3.5: The sender S sends the first packet on the flow containing the signed BC. Node X, upon receiving the packet, validates the capability by verifying the receiver signature and checking the route field for its information. If present, X forwards the packet. Y follows the same course of action and the packets arrive at R. Intermediate nodes cache the signed capability BC along with the next hop.

3.6.2 RAC: Protocol Design

In this section, we describe the operation of the RAC capability protocol. Specifically, we discuss the steps followed by the sender, the receiver, and individual nodes along the route to generate and repair capabilities. We also discuss the operation of RAC with regards to capability-enabled packets, before, during, and after a route change.

RAC Capability Request

- *S: Compute* $M_1 = (RFC, T_0)$
- $S \rightarrow R$: $\langle M_1 \rangle_{K_S^{-1}} + (RT_S)$

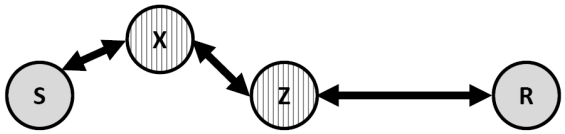

Figure 3.6: Node Y leaves the route, and the route between S and R is re-established along the route $\{S, X, Z, R\}$. The sender S issues a packet containing a normal capability to X. However, node X sees that Y is no longer the next hop, and thus modifies the packet to mark it an intermediate request, signs it, and appends the cached capability BC to the node Z. Node X also starts its grace period timer with a value of T_{GP}. Node Z first verifies that S possesses a valid capability using the EPIC protocol rules, even though it is not on the established route, before forwarding the packet and signed capability request to R. Upon receipt, R issues a capability to S along the route $\{S, X, Z, R\}$ with the original expiration time T_1.

The original request in this case is very similar between EPIC and RAC. The only addition is the route field RT_S, which is defined by the sender initially containing only the identifying information of the sender S. To create and maintain the route field $RT_{S,n}$, we use a cryptographically secure Bloom filter. Nodes need only be able to add elements (their own identifying bits) and check for their presence upon receipt [55–57]. The final value of the route field $RT_{S,n}$ - contains identifying bits for all nodes on the route. Other alternatives exist to identify the route, such as full source routing (by including identifying information such as IP addresses for all nodes on the route) or probabilistic node identification as illustrated in the SIFF approach [11].

RAC Capability Request Forwarding

- *N: Compute* $RT_{S,n} = RT_{S,n-1} + RT_n$
- $N \rightarrow R$: $\langle M_1 \rangle_{K_S^{-1}} + (RT_{S,n})$

With RAC, intermediate nodes, much like the fully route-dependent protocols, do not simply forward the packet. A node forwards the message after adding its identifying information to the existing route field $RT_{S,n-1}$, resulting in $RT_{S,n}$. Nodes cache the route

field $RT_{S,n}$ for use during future capability renegotiation, eliminating the need to maintain a route field for each packet. Nodes also cache the sender and receiver, the time of the request, and a flag indicating whether the request was generated by the sender or an intermediate node. This information is necessary to facilitate capability re-establishment without the participation of the sender.

RAC Capability Response

- *R: Compute* CH_n *and* CH_0
- *R: Compute* $BC = (T_0, T_{CV}, ID_S, ID_R, CH_0, RT_{S,R})$
- $R \rightarrow$ *S:* $E\{\langle BC \rangle_{K_R^{-1}}, CH_n\}_{K_S}$

The only difference in capability response between EPIC and RAC is the addition of the authorized route field $RT_{S,R}$, used to uniquely derive the capability BC. The respective public and private reference values CH_0 and CH_n do not change with the exception that CH_n takes as input the authorized route field $RT_{S,R}$. To support RAC's dynamic recovery and repair, the receiver includes the final established route $RT_{S,R}$ in the response. This value provides an authoritative basis for comparison for future capability-enabled packets, allowing for route-dependent forwarding decisions. Figure 3.7 provides an illustration of what is included in an RAC capability. These definitions of the common factors do not change between RAC and EPIC. The RAC protocol includes, in addition to the EPIC capability components, one additional parameter to support route adaptation, and that is the authorized capability route $RT_{S,R}$.

RAC Protocol Operation: Authorized Packets & Capability Repair

- *S: Compute Capability Hash Values (*CH_n*,* CH_{n-1}*, ...,* CH_0*)*
- *S: Compute Initial (First Packet) Capability* $CAP_0 = \{BC, CH_0, 0\}$
- *S: Compute Individual Capabilities* $CAP_i = \{CH_i, i\}$

- $S \rightarrow R$: *Data Packet* $\mid CAP_i$
- ***Intermediate Node*** N***, first packet of flow*** $(i = 0, RFC = 0)$***:***
  - N *verifies the receiver signature* S_R *and the base capability* BC
  - N *verifies its presence in the route field* $RT_{S,R}$
  - N *caches base capability* BC *and next hop and forwards the packet*
- ***Intermediate Node*** N***, subsequent packets*** $(i \neq 0, RFC = 0)$***:***
  - *Verify* $CH_{i-1} = f_h(CH_i)$*. If packet is outdated or a duplicate, drop the packet. Check that capability has not expired using* T_0 *and* T_{CV}*. If the route has not changed, forward the packet.*
  - *If the route has changed (the next hop differs from the cached value), mark the packet as a request (*$RFC = 1$*), sign the packet, and include the cached base capability* BC *and the route information* $RT_{S,n}$ *from the sender to the current node in the forwarded packet.*
    - $N \rightarrow R$: *Data Packet* $\mid \langle BC_{S,R} \mid RFC \rangle_{K_N^{-1}} \mid RT_{S,n}$
  - *Verify* $CH_{i-1} = f_h(CH_i)$*. If packet is outdated or a duplicate, drop the packet. Check that capability has not expired using* T_0 *and* T_{CV}*.*
- ***Intermediate Node*** N ***after route change*** $(i \neq 0, RFC = 1)$***:***
  - *If packet does not contain* BC *and* $RT_{S,n-1}$ *from previous hop, drop the packet.*
  - *If packet signature does not match the identified signing node, drop the packet.*
  - *If packet contains* BC *and* $RT_{S,n-1}$ *from previous hop:*
    - N *verifies* BC *is still a valid capability using* S_R*, verifying* $CH_{i-1} = f_h(CH_i)$*.* N *verifies that capability has not expired using* T_0 *and* T_{CV}*.*
    - N *caches the current route* $RT_{S,n} = \{RT_{S,n-1}, RT_n\}$*, the request time* T_R*, the source and destination* S *and* R*, the next hop, and the base capability* BC*.*

* *N starts its grace period timer T_{GP} for this flow and forwards the packet to the next hop with the current route $RT_{S,n}$.*

- ***Intermediate Node N, provisional packet (during capability repair):***
  - *(EPIC Verification): Verify $CH_{i-1} = f_h(CH_i)$. If packet is outdated or a duplicate, drop the packet. Check that capability has not expired using T_0 and T_{CV}.*
  - *N verifies that the grace period is still valid using the current time T_0, the repair time T_R, and the grace period T_{GP} such that $T_0 - T_R \leq T_{GP}$.*
  - *If the grace period is still active, N forwards the packet. If it is not, then the packet is dropped. No packets, request or data, are forwarded after the expiration of the timer T_{GP} for a given source and destination pair.*
- ***Receiver R after route change*** *($i \neq 0, RFC = 1$):*
  - *If R has an existing capability BC_0 for S:*
    * *If an identical request has been responded to more than the allowed number of times, drop the packet.*
    * *If R has not seen an identical request from S with route $RT_{S,R}$:*
      · *R: Compute CH_n and CH_0*
      · *R: Compute $BC = (T_0, T_{CV}, ID_S, ID_R, CH_0, RT_{S,R})$ where T_{CV} is the original expiration time from BC_0 and $RT_{S,R}$ is the current route*
      · *$R \rightarrow S$: $E\{\langle BC \rangle_{K_R^{-1}}, CH_n\}_{K_S}$*
      · *Cache request for S with route $RT_{S,R}$ and time T_0*

There is one key addition required for RAC to support to support dynamic recovery and reconfiguration that is otherwise irrelevant for true path-independent protocols - $RT_{S,R}$, the overall current route for the capability-enabled packet. This value is initially empty aside from the source's information whereas in its final form it represents all nodes on route between S and R.

The operation of RAC slightly differs from that of EPIC. In the case of EPIC, new nodes on the route would receive the authoritative capability value BC from its one-hop upstream neighbor. However, in RAC, if a node determines that it is not part of $RT_{S,R}$, that quantity would represent authorization for a different route. We define T_{GP} as the grace period for forwarding packets on new routes based on the configuration-time maximal approximate expected value for a packet's round-trip time in the network. Following this, the first node N that receives a capability-enabled packet containing an authorized route that recognizes a change in the route (a new next hop) will mark the packet an intermediate request, sign it, and forward it as provisionally authorized packet. Intermediate nodes only do this for a time period of T_{GP} (approximately $2 * RTT_X$) for each sender/receiver pair, which allows for the source to successfully establish, receive, and use the updated capability.

Intermediate nodes cache the current time, current route, and destination for each intermediate request. This holds whether an intermediate node signs a packet as an intermediate request or receives a packet previously signed by another node as an intermediate request. When the time period T_{GP} has expired, then the node will not forward the packet. This can happen for multiple reasons - the sender is no longer communicating with the destination, the route was not selected by the sender for use, or transmission delays occurred with the response packet. Receivers in RAC respond once to each intermediate request only, and will subsequently wait for a period of T_{GP} before responding to another intermediate request for the same source and route. Also, receivers will only respond to intermediate requests if they have previously issued a capability to that source that would otherwise still be active. An authenticated request (one signed by the original sender), however, will be responded to immediately regardless of the status of any other requests.

Protocol Maintenance: RAC Capability Renewal

Receivers in RAC can respond to requests generated by intermediate nodes on the route, but only if they have previously issued a capability to that source that would still be valid at the time of response generation. As with EPIC, the only maintenance required for

RAC capabilities is the periodic renewal either as capabilities approach the end of their validity period or intermediate capability renegotiation fails. The protocol interaction for maintenance between senders and receivers is identical to that of the initial negotiation.

Renewal caused by route changes in RAC is transparent within the capability validity period provided the flow is active, since an intermediate request can only extend the life of the capability until the end of the originally established route. RAC will only respond to requests generated by an intermediate node provided it has issued a capability to the proposed source in response to a signed request within that same time period. Intermediate nodes will only forward unauthorized packets for a limited time period T_{GP} and the destination will not respond to duplicate requests for the same time period more than a finite number of times. We note that intermediate nodes may forward multiple requests during the grace period provided the parameters are not identical to previous requests. Identical requests can be forwarded a finite number of times to allow for normal channel losses, collisions, and transmission errors. In all cases, however, the timer does not reset.

3.6.3 RAC Capability Components

BC: Base Capability	
ID_S	Sender ID
ID_R	Receiver ID
CH_0	Initial Capability Hash Value
T_0	Initial Capability Timestamp
T_{CV}	Capability Chain Validity Period
$RT_{S,R}$	Authorized Capability Route (RAC)
Receiver Signature S_R	
CH_i	Current Capability Hash Value

Figure 3.7: Components of a RAC Capability

- *Authorized Capability Route - $RT_{S,R}$:* For each capability, the source and destination will operate based on an established (authorized) route. Depending on the routing

protocol, there may be multiple capability requests, each with differing routes. When a node generates a capability request, it adds its own identifying information to the route field. Each node that forwards the request to the destination subsequently adds its own information to the route field. When the destination receives the request and generates a formal capability response, it will include the route of that request ($RT_{S,R}$) in the response. This will allow the sender to include the authorized route with each capability packet.

The route field can be implemented in a variety of ways. A Bloom filter provides a space-efficient mechanism for constructing route membership, and its supported operations (adding a member to the set and checking for a given element's membership in the set) meet the requirements for RAC's route field. While Bloom filters allow for false positives, this depends on the number of elements in the set, and in most mobile networks the number of nodes (and as such the route length) is sufficiently low enough that Bloom filters are a workable solution. With shorter routes, however, even simpler solutions - such as simply appending node address or ID information - are workable as well. We note that the actual implementation of the route field is orthogonal to the implementation of the RAC protocol as a whole.

3.6.4 RAC: Security Analysis

In this section, we evaluate the security of the RAC protocol with respect to various attacks. We use the defined notation from Section 3.4, while the protocol is again evaluated under the following assumptions:

1. The digital signature scheme and its underlying identity-based cryptosystem are secure.
2. The one-way hash function used in capability generation and verification is secure.
3. A node's private key is secure, preventing one node from masquerading as another.
4. Senders and receivers are not compromised at the time of capability establishment.

As RAC uses the same underlying mechanism as EPIC, the security analysis with regards to forgery, interception, replay attacks, data modification, and revocation as detailed in Section 3.5 apply. The structure of the RAC protocol, however, adds additional security characteristics - specifically, the structure of the route field RT and authentication of the route information warrants additional consideration.

Attacks on Capability Establishment

We note that as with EPIC, RAC is also susceptible to disruptions in legitimate packet forwarding. While the solutions proposed for EPIC also apply to RAC, additional complexity is introduced in that the correctness of a given route, potentially including its order, is important to the establishment and maintenance of RAC capabilities. The normal operation of RAC mandates that the route field RT be modified by each node routing the packet, so it is necessary to prevent malicious modification of this field.

Aggregate signatures, which represent the authentication of a single message or sequence by multiple independent nodes, can be used to help secure RAC against such attacks, both during the initial capability request phase and the capability repair phase. Ultimately, to provide effective security against such attacks, the aggregate signature must prevent malicious nodes from modifying the route information without detection.

Forward-secure sequential aggregate signatures have been proposed to solve this problem. Ma proposes a mechanism to allow signatures generated at different times with different keys on different messages to be combined into a single constant-size signature. Compromised nodes cannot truncate, delete, or modify aggregate signatures and message order is preserved [58]. Some mechanisms are designed to work exclusively on different messages, which is directly applicable to RAC [59], while others are required to work on the same message, which is directly applicable to EPIC [60]. The problem is somewhat well-studied and aggregate signatures have been proposed to secure routing protocols [61–63]. Some aggregate signature mechanisms are also designed with identity-based cryptography in mind, making them particularly attractive for RAC [59, 62].

Bloom Filters

We propose the use of Bloom filters to support a space-efficient implementation of the route field RT. For RAC, the route field RT is intended to represent the route information for a given route by storing bits securely corresponding to some subset of nodes in the network. Cryptographic security is necessary, as malicious nodes should be prevented from modifying the route information in an attempt to masquerade as one or more additional nodes [55]. With the known property that a Bloom filter with all bits set to one will always return a positive result when queried, simple threshold approaches (effectively limiting the number of elements that can be stored) can be used to prevent such manipulation [15].

Other variants of Bloom filters exist and may provide better security for RAC's intended purpose. Complement Bloom filters, for example, represent the inverse of traditional Bloom filters as they provide no false positives but finite false negatives, an acceptable alternative for a deny-by-default protocol. Yes-No Bloom filters also represent a significant improvement as they substantially reduce false positives by storing both the members of the set (an ordinary Bloom filter) as well as those elements that result in false positives. This approach can reduce false positives by nearly an order of magnitude while still maintaining the computational efficiency of a normal Bloom filter [64]. Ultimately, it is necessary for RAC to employ a secure variant of the Bloom filter to minimize or eliminate attempted modification of the route field RT.

Flooding and Remote Packet Injection

The RAC protocol differs from EPIC and RDC in two important ways. First, authorized packets can trigger capability renewal based on the current and established route. Second, the first packet in an authorized flow also includes a field containing route information, so each node can verify that it has participated in capability establishment. Capability renewal by intermediate nodes determines the difference in the route between the initial route established and the current route. In general, if the current node is not on the authorized route, the capability must be renewed along the current route.

With RAC, the combination of the use of a single authorized route and the ability of intermediate nodes to request capability renewal potentially allows a receiver to identify anomalous behavior. Consider, for example, the case where a valid route $RT_{S,R}$ is active and authorized. A malicious node on that route re-routes the packet to a remote location will then either fail to reach the destination (in which case the grace period T_{GP} prevents significant impacts) or generate a capability response from the destination along the malicious route. A node may assume that its route has been changed and attempt to use the new capability. However, even if the route has not actually changed, this subsequent packet will not generate a capability request along the original route it had just abandoned unless the node responsible for diverting the packet signs a new intermediate request, effectively incriminating itself. If it does generate a request along the original route, thrashing can be detected and with high probability we can assume that a remote packet injection attack is underway. The node can resume operation with the original capability, if available, or request a new capability altogether. We illustrate such an attack in Figure 3.8.

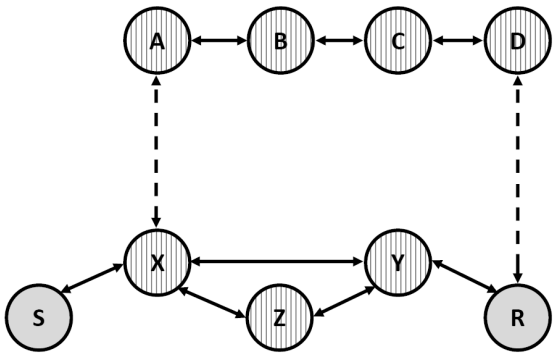

Figure 3.8: The sender S and the receiver R have established authorized communication on the route $RT = \{S, X, Y, R\}$ using capability BC_0. Node X then forwards a packet to A using a wormhole and marks it as a request. A, verifying the capability as accurate, starts its timer T_{GP} and forwards the packet to B. The process continues until R is reached, at which point a new capability BC_1 is issued along the route $RT = \{S, X, A, B, C, D, R\}$.

With RAC, the normal operating nature of a MANET means that capabilities will be dynamically reestablished based on the tracking of a given packet's flow. If it deviates

from the established route of the current capability, intermediate nodes forward the packet provisionally and mark it as a new capability request. Based on this knowledge, malicious nodes could theoretically request capabilities for injected packets, triggering a response from the destination to the sender. RAC, however, achieves a significant level of protection against such packet injection attacks, since it only allows for a single active capability between a source and a destination. There are several factors limiting the scope of such an attack:

- **Duplicate Prevention**: Intermediate nodes may, during an attack, forward multiple requests from previous intermediate nodes. The receiver R, however, will not respond to duplicate intermediate requests as determined by the source S, the receiver R, and the proposed route $RT_{S,R}$ more than a finite and determined number of times. While R cannot necessarily determine the unique nodes comprising a proposed route, it can determine whether the value $RT_{S,R}$ is a route for which it has already issued a response. R is also able to determine the signing node N, and duplicate requests are only retransmitted a finite number of times to accommodate normal network failures.

- **Sender-Controlled Use**: Senders retain full autonomy over which capabilities they use. In the event a remote packet injection occurs and an unneeded (malicious) capability is generated by the receiver, the sender does not have to use it. Consider the following situation:

  - S has a capability BC_0 for the receiver R.
  - Intermediate node X attempts to launch an injection attack. It must sign a new capability request for the proposed repaired route as shown in Section 3.6.2.
  - The receiver R issues capability BC_1 for the new route.
  - S receives the new capability BC_1. If S uses BC_1, it will only be accepted along the route field $RT_{S,R}$ defined in BC_1. As X cannot forge capability values or force S to use its original capability BC_0, it cannot attempt a multipath attack without signing a request for another repaired route.

In this case, the sender will be in possession of two legitimate capabilities, but only one of them can be used. Intermediate nodes only cache a single active capability value and as such use of any renewed or repaired capability invalidates previous capabilities. For each node, including the destination, only a single capability can be active at any point in time. If a node does not use a malicious capability, its original capability remains valid, while the new capability will not be used. In Figure 3.8, capability BC_0 is not invalidated until BC_1 is used. In this case, the attack is reduced to a route manipulation attack, which is ultimately an attack on the routing protocol rather than the capability mechanism. As a failsafe, if the normal operation of the RAC protocol does not result in a successful repair, a new capability request signed by the sender S will be forwarded by intermediate nodes and responded to by the receiver R, so the sender retains ultimate control over the repair process.

- **Self-Incrimination**: Since intermediate nodes must sign requests when they determine a route has broken, they effectively identify themselves as malicious if they forward the packet to a remote location. The only way subsequent nodes will forward either provisional authorized packets or intermediate capability requests is with a previously received valid signature, which would serve to identify the node that initiated the wormhole attack. If a node X launches a wormhole attack which causes a legitimate sender to use an alternate route, security is preserved as if a node X forwards a packet containing the new capability to a subsequent node, it will not be able to verify the hash value CH_i since its cached capability value belongs to a different hash chain. Similarly, if X tries to re-establish the capability along the original route, it will identify itself as the likely source of the wormhole attack. It also follows that if S uses the subsequent capability, it will invalidate the capability established along the wormhole route.

- **Temporal Forwarding Limitation**: Both intermediate nodes and destination nodes do not allow for unlimited requests. An intermediate node will only forward a duplicate intermediate request a finite number of times. Similarly, intermediate nodes

only forward provisional packets for a limited time T_{GP}. Destinations can limit the number of route changes they will allow within a given capability validity period and only respond at fixed intervals or a fixed number of times; for RAC a destination will not respond to a given intermediate request more than once every T_{GP} seconds. Intermediate nodes allow for multiple potential route changes during capability re-establishment, but the timer does not reset following the initial repair request.

3.7 Performance Results

3.7.1 Simulation Methodology

We use simulation models to compare the operation of the EPIC, RAC, and RDC protocols. We have three main goals for simulation: first, to show that EPIC's performance with regards to denial-of-service prevention is at least statistically equivalent to that of traditional route-dependent capabilities; second, to evaluate the performance and efficiency impacts of EPIC's flexible enforcement mechanisms on the network; and third, to evaluate EPIC's resilience to attacks under a variety of scenarios. For purposes of comparison, route-dependent capabilities are implemented by including route information in the packet headers and comparing them with the established route, which is included in the capability. When a route breaks, the capability is considered broken and a new request must be issued by the source. We use the methodology of independent experiment replications to obtain simulation result within a 95% confidence interval and validate our results [65].

3.7.2 Simulation Framework and Scenarios

We use QualNet versions 5.0, 7.1, and 7.3. QualNet is a commercially available network simulator based on UCLA's GloMoSim project, a scalable Parsec-based parallel discrete-event simulator suitable for mobile ad hoc networks [66]. To evaluate the potential security, performance, and efficiency aspects of EPIC and RAC, we simulate using both the AODV and OLSR routing protocols and a variety of mobility models and traffic patterns. Details

and simulation parameters are listed in Table 3.1.

Table 3.1: Summary of Simulation Scenarios

Parameter	**Values**
Routing Protocols	AODV, OLSR
Mobility Models	Manhattan, MMTS
MMTS Models	City, Rural, Urban
Number of Nodes	50 (Manhattan)
	158 (MMTS Rural)
	250 (MMTS Urban / City)
Simulation Area	2.25 km^2 (Manhattan)
	9 km^2 (MMTS)
Applications	FTP, CBR
Simulation Time	1800 s
Number of Runs	20
Confidence Interval	95%

Routing Protocols

The Ad Hoc On-Demand Distance Vector routing protocol (AODV) is a reactive protocol that only establishes routes when it needs to support communication between endpoints. The Optimized Link State routing protocol (OLSR) is a proactive protocol that maintains optimal routes by establishing and maintaining awareness of the entire network's topology.

Mobility Models

We use the Manhattan and MMTS mobility models, derived from the Generic Mobility Simulation Framework (GMSF). These models are described in detail in [67] and [68]. Characteristics of the mobility models are described below.

- Manhattan: This model is designed to simulate vehicle and pedestrian traffic in a city setting. Nodes move along a restricted grid-like pattern at randomly determined speeds and either turn at the intersection of horizontal and vertical grid lines (spaced at 200 meter intervals) or continue straight; decisions regarding movement are not

made except at defined intersections. Manhattan mobility simulations are conducted with 50 nodes in a 1500 m x 1500 m (2.25 km^2) area. The intersection distance is a uniform 200 meters, and node movement speed is between 12 m/s and 14 m/s with acceleration distributed between $\{-0.1, 0.1\}$ m/s^2. For a given node, the movement speed is limited by leading nodes. Manhattan models cover 41.2% of the simulation area and have relatively low node density.

- MMTS (General): MMTS, The Microscopic Multi-Agent Traffic Simulator, is designed to simulate realistic vehicular movement patterns over real-world regional road maps in Switzerland. Because they are limited to roads, nodes cover at most 9% of the simulation space, which is 3000 m x 3000 m (9 km^2). Following this, maximum node density is much higher. The distance between nodes is variable, but most frequently ranges from a uniformly distributed range of 500 meters to 2000 meters. The average speed of vehicles is about 10.5 m/s, but this is not uniform as speeds are clustered at or around two peaks: 3 m/s, which represents traffic, and 12 - 17 m/s, which represents movement at normal speed. As traffic is limited to defined roads, routes are longer on average.

- MMTS (City): In this model, the road map covers a small portion of the map, but roads are in close spatial proximity and thus node density is potentially very high. Simulations are conducted with 250 nodes, so the potential for congestion (clustering) is also very high. Following this, movement speed is the lowest of the MMTS models.

- MMTS (Urban): In the Urban model, roads are approximately evenly spaced and cover more of the map than other MMTS models. Movement speed and node density are both moderate, with the probability of both high movement speed and node congestion minimized. Simulations are conducted with 250 nodes and this model is ideal for evaluating performance under balanced conditions.

- MMTS (Rural): The Rural model covers a minimal area of the map, and represents multiple high-speed roads with minimal intersection points. Node density is lowest and thus the possibility of network connectivity issues is highest. Additional, average movement speed is maximized. Simulations are conducted with 158 nodes (the maximum allowable by the GMSF) and represents the most difficult network environment despite the fact that the risk of congestion is minimal.

Based on the characteristics of our routing protocols (the reactive AODV and the proactive OLSR), we can intuitively expect certain performance levels based on the mobility model. However, we note that both AODV and OLSR were evaluated during the design of the GMSF, and the author's conclusion was that the performance of a given protocol is highly dependent on the mobility model [67]. The overall characteristics of the models are given in Table 3.2.

Table 3.2: MMTS Model Characteristics

	Node Density	**Map Coverage**	**Mobility Speed**
Manhattan	Moderate	High	Moderate
MMTS City	Very High	Moderate	Low
MMTS Rural	Low	Very Low	High
MMTS Urban	Moderate	Moderate	Moderate
	Direction Difference	**Speed Difference**	**Map Area**
Manhattan	Moderate	Very Low	Low
MMTS City	Moderate	Low	High
MMTS Rural	High	High	High
MMTS Urban	Low	Low	High
	Node Distance	**Node Reachability**	
Manhattan	Moderate	Very High	
MMTS City	Low	Moderate	
MMTS Rural	High	Low	
MMTS Urban	Moderate	High	

Applications

We also conduct simulations using both CBR and FTP transport-layer applications. For CBR, one node acts as the CBR client and one node acts as the CBR server. Two identical sessions are established to facilitate easier capability establishment and maintenance (two nodes participate, with each node acting as both client and server). The client sends packets for a 10 minute interval with a 250 ms inter-packet delay for a total of 2400 packets. This represents relatively low channel utilization over a long period of time, allowing the effects of mobility greater significance. For FTP, one node acts as the FTP server and one node acts as the FTP client. The server sends 1000 packets, each 1000 bytes in length, to the client continuously until the session is complete. This represents maximal channel utilization over a potentially long period of time, allowing varying degrees of significance to both channel contention and mobility.

Metrics

For CBR applications, performance is studied through the end-to-end delay metric. Since packets are not being transmitted constantly but rather at specific intervals, end-to-end delay represents a more accurate approximation of performance for typical CBR applications (such as voice, audio, and video data). We also include the variance in the end-to-end delay, which is commonly known as jitter. This helps to establish some level of confidence in the reported end-to-end delay. For FTP, the performance metric is straightforward and is represented by the throughput. The application transmits data at maximum effort until the session is closed, and as such the total data and total time are appropriate metrics. Efficiency for both applications is defined as the "goodput" percentage of traffic between a sender and a receiver, measuring the capability protocol-related overhead data against the actual application data.

EPIC, RAC, and RDC Control Overhead Information & Computational Delays

We define the following packet control information for all of our simulated models.

Table 3.3: Capability Request: EPIC (20 Bytes)

Data Field	**Bit Width**
RFC	4
Reserved Control	4
Request Timestamp T_0	24
Sender Signature S_S	128

Table 3.4: Capability Request: RAC / RDC (36 Bytes)

Data Field	**Bit Width**
RFC	4
Reserved Control	4
Request Timestamp T_0	24
(Optional) Sender Signature S_S	128
Authorized Route $RT_{S,R}$	128

Table 3.5: Capability Response: EPIC (96 Bytes)

Data Field	**Bit Width**
RFC	4
Reserved Control	4
Request Timestamp T_0	24
Sender ID ID_S	32
Receiver ID ID_R	32
Capability Chain Validity Period T_{CV}	32
Capability Hash Chain Initial Value CH_n	256
Capability Hash Chain Anchor CH_0	256
Receiver Signature S_R	128

EPIC, RAC and RDC all incur computational delays due to their use of digital signatures (both generation and verification) as well as hash functions. Receivers and senders make extensive use of all types of cryptographic functions, while intermediate nodes primarily need to verify hash functions. With RAC, intermediate nodes do sign requests.

Table 3.6: Capability Response: RAC / RDC (112 Bytes)

Data Field	Bit Width
RFC	4
Reserved Control	4
Request Timestamp T_0	24
Sender ID ID_S	32
Receiver ID ID_R	32
Capability Chain Validity Period T_{CV}	32
Capability Hash Chain Initial Value CH_n	256
Capability Hash Chain Anchor CH_0	256
Receiver Signature S_R	128
Authorized Route $RT_{S,R}$	128

Table 3.7: Capability Data Packets: EPIC / RAC / RDC (36 Bytes)

Data Field	Bit Width
Flow ID	16
Sequence Number i	16
Current Capability Hash Value CH_i	256

It is difficult to assign a computation delay to each specific operation. The expected computation time is highly dependent on the specific algorithm being used as well as the underlying hardware being used. In the established literature, results vary widely; signature operations can range from less than 0.1 ms on modern hardware to greater than 10 ms on miniaturized sensor hardware [69–71]. The same holds true for hash operations [72]. We note that while the true numbers are highly dependent on both the specific algorithms and the underlying hardware, the relative performance effects are unchanged.

3.7.3 Zero-Attack Throughput & Efficiency: AODV

Given the lightweight nature of EPIC, we expect that it will provide the highest potential performance for normal network operation. This is because of the minimal computation

Table 3.8: Computation Overhead: EPIC / RAC / RDC

Operation	Computation Time
Hash Function	0.5 ms
Digital Signature Generation	20 ms
Digital Signature Verification	60 ms

involved in verifying capabilities. Similarly, we expect RAC to be less complex than verifying each step of a route as we see in RDC, but it does have additional overhead due to the complexity of maintaining active route information. We would expect that the capability maintenance-related delays would be somewhere between EPIC and RDC, since its incidence of capability renegotiation nearly approaches that of RDC but the performance impacts are substantially reduced.

FTP

Figure 3.9 illustrates the FTP performance of all capability protocols with respect to multiple mobility models. Following this, Figure 3.10 illustrates the overhead efficiency for FTP applications when AODV is used as the routing protocol. The results are mostly straightforward; EPIC outperforms RAC while both outperform RDC. However, the mobility model has a much greater impact on efficiency than does the capability model.

We see that more permissive the capability model is, the higher the throughput, an intuitive result since they result in fewer capability-related disruptions to the network. One important thing to note is that the difference between the various capability models (EPIC and RDC) is not significant compared to the difference between mobility models. Also, the difference between EPIC and RAC is even smaller. While the network density might help reduce the delay in route re-establishment, these benefits can be offset by the additional overhead incurred in both sparse (Rural) and dense (City) networks. As the mobility model leads to higher overhead, especially capability-related overhead, both performance and efficiency decline. The results show that multiple factors interact to influence both

performance and efficiency.

We also evaluate the performance of EPIC against AODV without any modifications, and we see that for the Manhattan mobility model EPIC incurs a 7.3% penalty to throughput. While this is an expected result as the capability mechanism necessarily incurs additional computational and networking overhead, the performance penalty is relatively small.

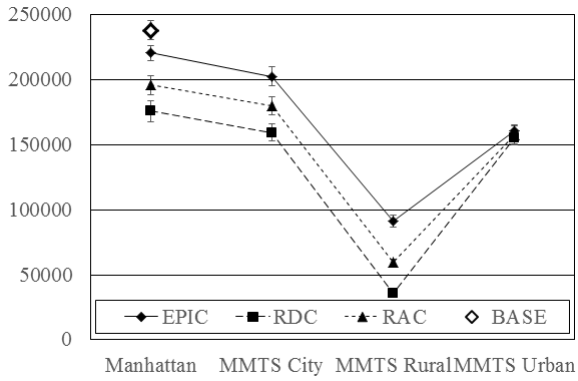

Figure 3.9: Average AODV / FTP Throughput (kbps)

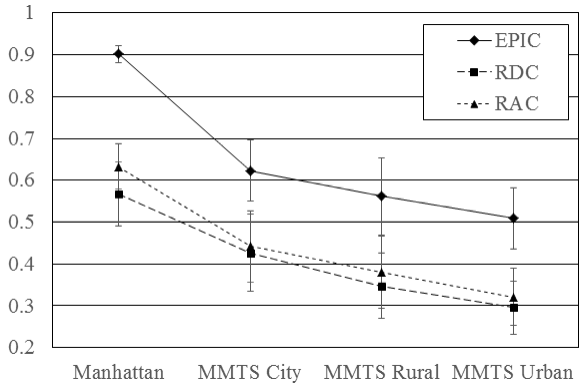

Figure 3.10: Average AODV / FTP Overhead Efficiency

CBR

Figures 3.11 and 3.12 illustrate the CBR performance for each capability model using AODV as the routing protocol. Unlike the FTP results, CBR performance is far more likely to be stable as a long-running application. While we see some of the same factors - in particular, EPIC has the highest performance, due mainly to the reduced incidence of capability renegotiation - there are some important differences. The mobility model in particular has a reduced effect compared to the capability verification method, having approximately the same impact as the capability model. Contrast this with FTP, where the significant variation comes from the mobility model with relatively little contribution from the capability model. The variance in the end-to-end delay tracks closely with the reported end-to-end delay, and ranges from approximately 16.0% of the reported delay for EPIC to 22.7% and 22.6%, respectively, for RDC and RAC.

Figure 3.13 illustrates the overhead efficiency for the CBR application for each capability model and routing protocol. We see here some variation with respect to the routing protocol. The capability model has a much more substantial effect than the mobility model. While the most intuitive result still holds true (the more permissive or efficient a capability model is, the more the incidence or impact of capability renegotiation decreases, in turn resulting in lower overhead), it is difficult on the whole to predict what will ultimately be the dominating factor. However, it is important to see that the degree of control a network member holds over its mobility model or routing protocol is very low (and potentially zero), where nodes could potentially negotiate the use of a different capability model. It follows then that the impact of the capability model is of greater importance than the mobility model and the routing protocol even if significance is reduced.

The performance and efficiency are, as with FTP, affected by the mobility model in intuitive ways. Also, as the FTP results show, it is apparent that the effects of the mobility model cannot be simply quantified or correlated directly to a single factor. Instead, multiple factors influence the final result. We note also that the base AODV performance for the Manhattan mobility model is 23.9% higher than EPIC, though in absolute terms the

inclusion of the EPIC security mechanisms adds only an additional 0.029 s to the average end-to-end delay. This again represents a relatively small performance penalty.

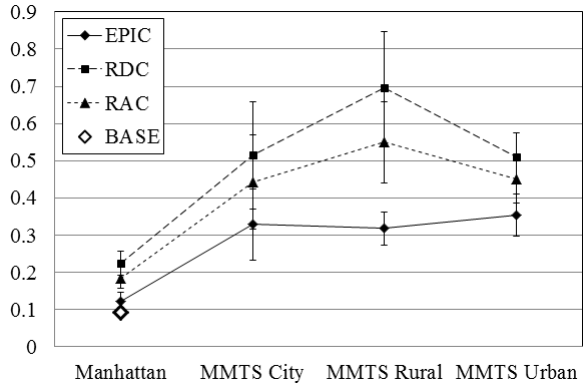

Figure 3.11: Average AODV / CBR End-to-End Delay (s)

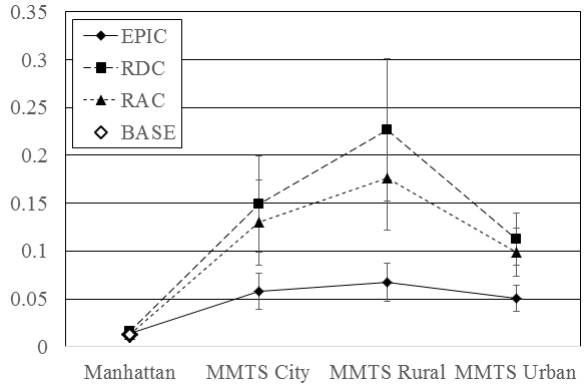

Figure 3.12: Average AODV / CBR Jitter (s)

3.7.4 Zero-Attack Throughput & Efficiency: OLSR

As with AODV, we expect that EPIC, which requires the least additional communication from a capability perspective, will offer the highest performance. However, as all protocols are able to take advantage of the proactive nature of OLSR, we do not expect a significant

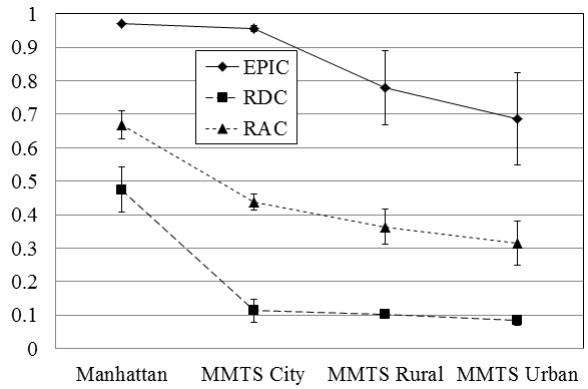

Figure 3.13: Average AODV / CBR Overhead Efficiency

difference between them. This is because of the minimal differences in capability operation given the route establishment and maintenance is handled proactively by the routing protocol - EPIC allows authorized communication as long as a route exists. RAC allows authorized communication as long as a route exists, and RDC must re-establish a capability along the route before allowing authorized communication. As before, we also expect that the performance of the RAC protocol will fall between EPIC and RDC, but given the nature of OLSR the differences should not be substantial.

FTP

Figure 3.14 illustrates the FTP performance of all capability protocols with respect to multiple mobility models. Following this, Figure 3.15 illustrates the overhead efficiency for FTP applications when OLSR is used as the routing protocol. The results are mostly straightforward; EPIC outperforms RAC while both outperform RDC. The mobility model, however, has different effects on OLSR than it does on AODV. The MMTS City model, with its maximum density, likely incurs congestion-related delays due to the very high network load caused by the routing protocol. Research supports this conclusion, as OLSR has been shown to scale poorly to larger and denser networks [73, 74]. The combination of moderate congestion and density along with similar node direction makes the MMTS Urban model

an ideal performance evaluation.

With OLSR, we cannot evaluate the peripheral statistics as we do with AODV since the protocol itself proactively maintains routes regardless of whether nodes wish to communicate or not. This means that capability maintenance does not have much of an effect as it is dependent only the end-to-end delay of the network, which is almost identical among capability protocols. Also, as OLSR is proactive, the network load from the routing protocol is substantially higher than from the capability protocol. This dominates capability protocol-related overhead and ultimately has considerably more impact on performance.

The performance of OLSR significantly exceeds that of AODV. This result is supported by much of the existing research, which has shown that OLSR outperforms AODV in terms of both throughput and delay, sometimes by orders of magnitude [74–81]. Similarly, other aspects of the protocols are apparent - AODV, for example, performs poorly as mobility increases [73, 74, 77, 82], while OLSR can exhibit performance impacts in dense networks [73, 74, 83]. We do not necessarily expect a straightforward result in terms of the performance differences between OLSR and AODV as multiple factors - mobility, routing protocol, application, and capability protocol - interact to affect performance and efficiency in different and sometimes conflicting ways.

We also evaluate the OLSR protocol using the Manhattan mobility model without any security enhancements, and we see again that the base performance of the routing protocol, as expected, exceeds the performance of EPIC. However, the proactive nature of OLSR substantially limits the performance effects of EPIC, RAC, and RDC; the inclusion of the EPIC protocol reduces performance by only 1.1%.

CBR

Figures 3.16 and 3.17 illustrate the CBR performance of each capability protocol and mobility model when OLSR is used as the routing protocol. Similarly, Figure 3.18 illustrates the overhead efficiency for CBR applications when OLSR is used as the routing protocol. The results with regards to both delay and jitter show little absolute difference between any

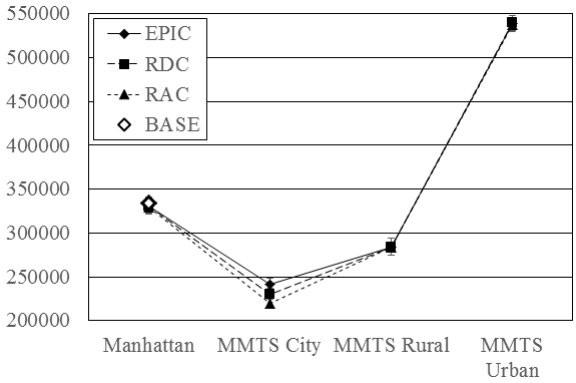

Figure 3.14: Average OLSR / FTP Throughput (kbps)

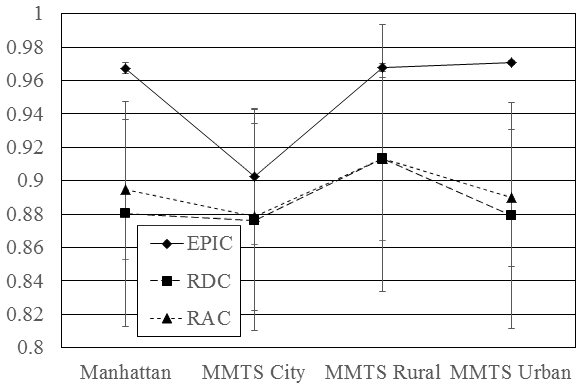

Figure 3.15: Average OLSR / FTP Overhead Efficiency

variable, whether we are referring to the capability protocol or the mobility model. The absolute value of the jitter is low, and as a percentage of the reported end-to-end delay the results are similar to AODV, ranging from 13.7% for RDC to 27.2% for EPIC. RAC is closer to the lower end at 16.7%.

The proactive nature of OLSR contributes to this result as it leads to fairly significant resilience to different mobility patterns. We note that previous research has established this as well [75–77, 79, 81, 83]. The mobility model, unlike with FTP, has little effect on OLSR. The ability of OLSR to maintain routes leads to lower delay as a route is almost

always available when requested. We note that in absolute terms the difference between both mobility models and capability protocols is low.

We note that the base OLSR performance for CBR with the Manhattan mobility model is 19.6% higher than EPIC. As with FTP, however, the performance impact in absolute terms is quite small as the inclusion of the EPIC security mechanism adds only an additional 0.009 s to the average end-to-end delay.

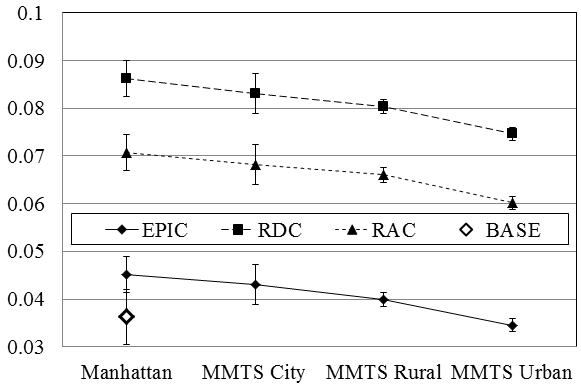

Figure 3.16: Average OLSR / CBR End-to-End Delay (s)

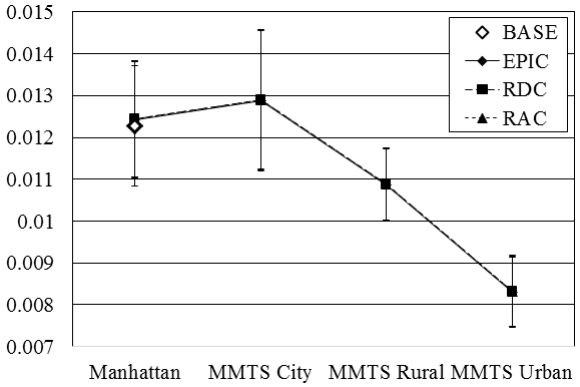

Figure 3.17: Average OLSR / CBR Jitter (s)

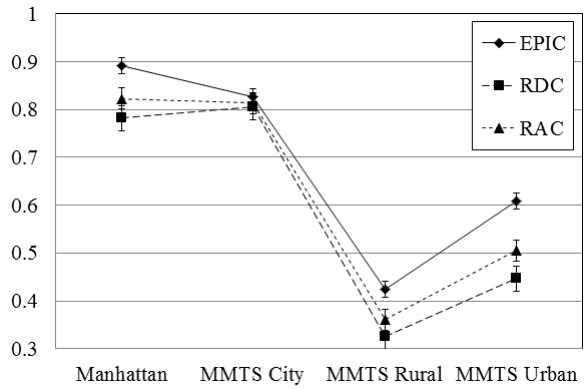

Figure 3.18: Average OLSR / CBR Overhead Efficiency

3.7.5 EPIC and RAC: DoS / DDoS Mitigation

As network capabilities are designed to mitigate DoS attacks in particular, it is necessary to evaluate their performance under such scenarios. For this simulation, we use the Manhattan mobility model using the AODV and OLSR routing protocols. We also use both CBR and FTP as the base applications. For FTP, we use a highly structured session that sends exactly 1000 items of 1000 bytes in size. For the attack model, we assume that malicious nodes are only able to flood unmodified capability-enabled messages. For each scenario, we define the value x as the total number of malicious nodes in the network. This is defined on the x-axis in each performance figure. In general, if the route is presumed valid, a given capability is accepted (or, in the case of RAC, a new request issued) once the next value in the chain is accepted. Thus malicious nodes are limited by the legitimate sending activity of the source.

For each scenario, we define one node as the primary (legitimate) sender S and one node as the receiver. The number of malicious nodes can be in the set $\{0, 1, 2, 4, 8, 16\}$, and the location of each malicious node is not fixed but rather dictated by the mobility model. We assume that a wormhole link exists between the source node (the legitimate sender S) and each malicious node, and the point of injection is the location of the respective malicious

node. The destination for all malicious packets, regardless of origination, is the originally defined receiver R.

Figures 3.19 and 3.20 illustrate the CBR performance of EPIC under various attack scenarios (different numbers of malicious nodes). We also compare the results to both RAC and RDC results. For all models, throughput degrades as the legitimate source contends with malicious nodes for limited resources. In the case of CBR, both RAC and RDC suffer little to no effect from malicious activity while EPIC takes a substantial performance hit as the number of attackers increases. As shown in Figures 3.21 and 3.22, the same is true of FTP (albeit to a lesser degree). In general, RAC is highly resistant to attacks. EPIC holds a significant advantage in low-attack networks, but degrades substantially as the number of attackers increases. RDC is highly resistant to attack, but also has the lowest performance in low-attack networks. RAC provides a solid balance, as it is highly resistant to attack and can operate at a consistent level even in attacker-saturated networks. From a zero-attacker to a 16-attacker network, RAC on average retains 84.3% of its base throughput compared to only 28.2% for EPIC. In fact, it only takes one attacker conducting packet injection attacks for RAC to surpass EPIC's performance.

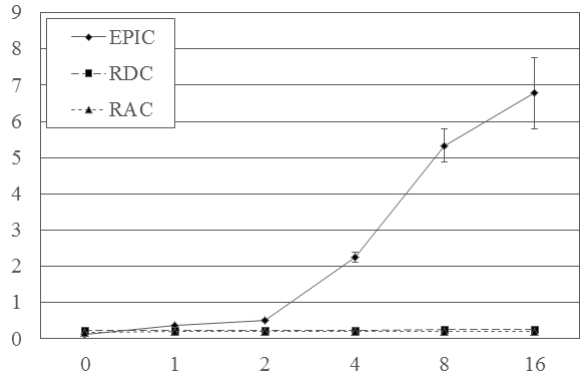

Figure 3.19: Average AODV / CBR End-to-End Delay (s) - DoS Resistance, Performance vs. Number of Attackers

Similarly, Figures 3.23, 3.24, 3.25, and 3.26 illustrate the effects of malicious nodes on

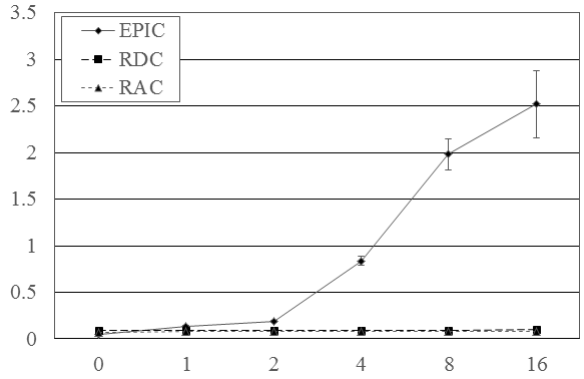

Figure 3.20: Average OLSR / CBR End-to-End Delay (s) - DoS Resistance, Performance vs. Number of Attackers

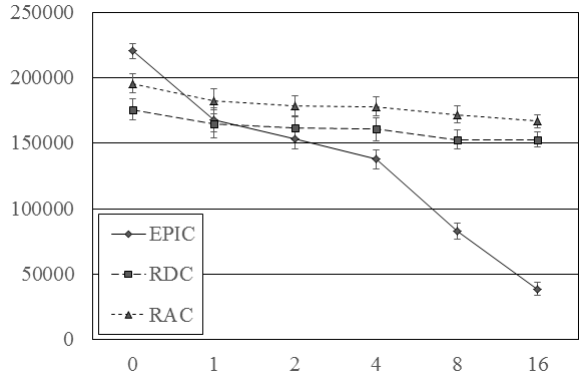

Figure 3.21: Average AODV / FTP Throughput (kbps) - DoS Resistance, Performance vs. Number of Attackers

efficiency. Again, we see that EPIC's enhanced security model (RAC) is highly resistant to attack, retaining 92.0% of the base (low-attack) efficiency compared to 39.4% for EPIC. There is little substantial difference between applications and routing protocols in terms of the relationship between efficiency; capability models perform very similarly for all combinations of routing protocols and applications. The results are very similar to performance in that EPIC performs best in low-attack networks while the more restrictive methods are more resistant to attack. As expected, RAC is flatter in the sense in that it does not exhibit

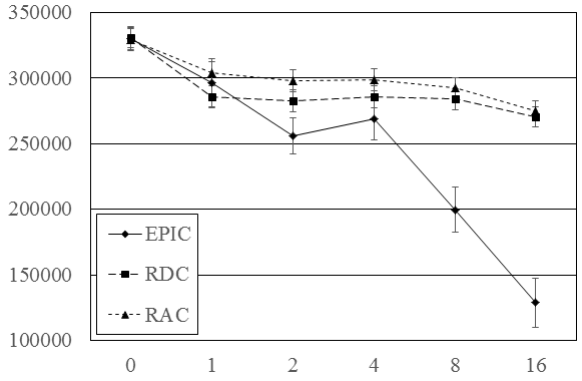

Figure 3.22: Average OLSR / FTP Throughput (kbps) - DoS Resistance, Performance vs. Number of Attackers

significant variation with regards to the number of attackers.

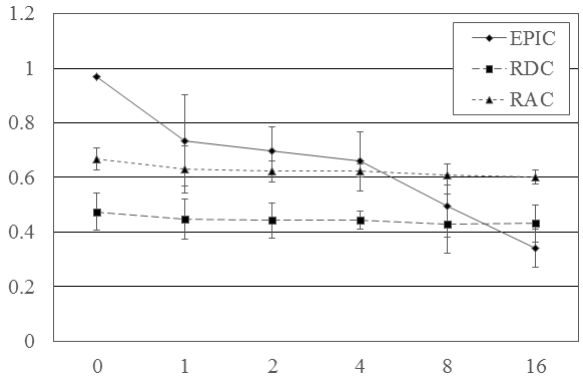

Figure 3.23: Average AODV / CBR Overhead Efficiency - DoS Resistance, Efficiency vs. Number of Attackers

3.8 Summary

In this section, we have presented EPIC, Efficient Path-Independent Capabilities, which represents a significant improvement in efficiency and performance on the existing unicast capability methods. We also present an enhanced security version of the mechanism, RAC, a

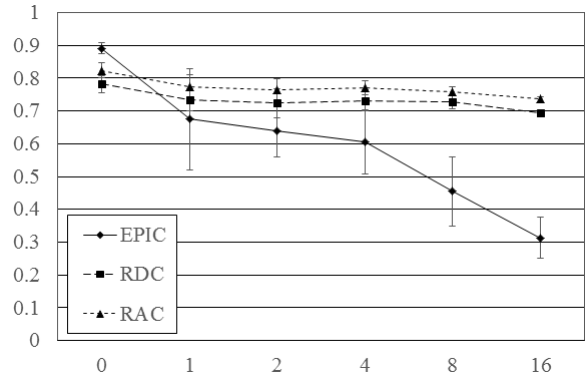

Figure 3.24: Average OLSR / CBR Overhead Efficiency - DoS Resistance, Efficiency vs. Number of Attackers

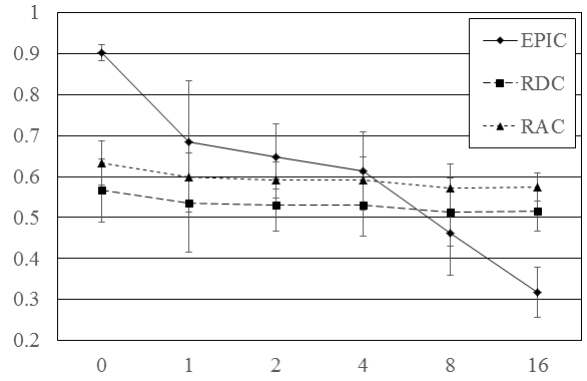

Figure 3.25: Average AODV / FTP Overhead Efficiency - DoS Resistance, Efficiency vs. Number of Attackers

capability mechanism which dynamically reestablishes capabilities along new routes, which represents an efficient and reasonably well-performing attack-resistant capability mechanism. EPIC is based on two key principles: reverse-disclosure hash chains, which allow two communicating entities to efficiently maintain secure communications over multiple time periods on a unique per-packet basis, and identity-based cryptography, which allows us to avoid the problem of public key exchange by having digital signatures and authentication methods dependent on a unique identity (such as a network address). These allow for

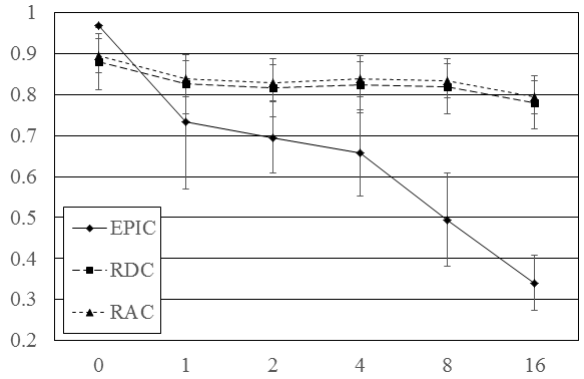

Figure 3.26: Average OLSR / FTP Overhead Efficiency - DoS Resistance, Efficiency vs. Number of Attackers

the most important aspect of EPIC - universal verification ability. This is precisely what makes both complete path independence and dynamic reconfiguration possible. Results indicate a statistically significant increase in performance and a significant reduction in routing-associated and capability-associated overhead. However, both the performance and efficiency of EPIC degrade rapidly as malicious nodes conduct authorized flooding attacks, injecting otherwise authorized packets at random locations in the network. This led us to develop the enhanced security version of EPIC, which builds on EPIC by utilizing dynamic capability repair to facilitate transparent maintenance and efficient operation. RAC allows for changes to the route while streamlining reconfiguration in such a way that allows legitimate traffic while mitigating the effects of malicious traffic while simultaneously remaining relatively transparent to the nodes involved. Further, we have shown that EPIC is not difficult to implement and represents a more flexible and efficient implementation of capabilities as compared to fully route-dependent capability mechanisms.

We have illustrated how EPIC can be used in conjunction with multiple different routing protocols and have simulated two very different protocols - AODV and OLSR. Simulation results show that EPIC provides as much as a 27.4% increase in performance and a 33.7% increase in efficiency over route-dependent capabilities (RDC), while RAC provides as much

as an 8.2% increase in performance and a 13.6% increase in efficiency over RDC. When compared to applications that do not employ any capability mechanisms, the performance impacts of EPIC and RAC are acceptable - for EPIC, FTP throughput is reduced by an average of 4.2%, and for RAC, FTP throughput is reduced by an average of 9.6%.

Chapter 4: Supporting Deny-by-Default for Multicast Traffic in MANETs

In this chapter, we propose a method to support deny-by-default communication for multicast traffic: EPIC-M (Multicast), a method that, like EPIC, combines reverse-disclosure hash chains, identity-based cryptography, threshold cryptography, and hop-by-hop verification to support the establishment and maintenance of effective multicast capabilities. We evaluate the performance of EPIC-M through simulations, including multiple mobility models and multicast routing protocols. We also analyze the security of the EPIC-M protocol.

4.1 Multicast Capabilities: An Overview

Multicast routing protocols employ groups that have an open access policy. Any node can send to the group by sending to the well-known multicast group address. Similarly, any node can join the multicast group, entitling them to receive communications addressed to that group. We focus on the problem of enforcing deny-by-default for multicast traffic by controlling sending access to the group. While multicast capabilities could be separated into distinct sending and receiving capabilities, we do not consider receiving capabilities to be a problem from a denial-of-service perspective. Also, restricting which nodes are able to receive can be accomplished using the standard approach of encrypting transmitted data with a group key that is only known to legitimate members of the group.

Most multicast routing protocols use group leaders or controllers to manage membership and group key distribution. We take the approach that a capability group controller (CGC), potentially separate from the multicast group controller, will provide the separate function of issuing and authorizing sending capabilities for the group. This only increases complexity

minimally as multicast group leaders are configured when the group is set up, usually offline, so all controllers that handle network management functions will be designated at the same time. We assume that all nodes that participate in multicast routing will forward packets that contain the necessary authorization from a controller. Since each node obtains its own sending capability independently, a necessary disadvantage of this approach is that multicast router nodes will have to maintain some state for each sender in a group.

To provide resilience to communication failures caused by node mobility or membership changes, group capability controllers can be configured to have multiple independent nodes that act cooperatively to issue capabilities. We adapt an approach based on k, n threshold signatures to provide resilience to such problems. Previous research has shown that threshold cryptography can aid in securing MANET communication against individual node failures [42, 43, 47]. In EPIC-M, senders create their own capabilities, but they are not valid until k partial signatures are obtained from the individual group controllers.

EPIC lends itself naturally to the problem of deny-by-default for multicast traffic for three main reasons. First, capabilities are not tied to routes, but instead to a logical multicast group. Not only will routes change frequently but in the case of a multicast, there are multiple paths that each packet will traverse. Second, the reverse hash chain mechanism is efficient for verifying a capability. Finally, the capability protocol is designed to operate independently from the routing protocol, which allows for both tree-based (e.g. MAODV) and mesh-based (e.g. ODMRP) multicast routing protocols to be used without requiring any changes to the capability protocol.

4.2 EPIC-M: Multicast Capabilities

4.2.1 Initial Bootstrapping

At network launch time, an offline authority outside of the network is assumed to be responsible for both the designation of the threshold (capability group controller) nodes and the initial creation of the multicast groups.

Following boot strapping, we can assume the following state for a given multicast group:

1. A $\{k, n\}$ threshold cryptography mechanism, acting as a capability group controller (CGC) for issuing capabilities, has been initialized among the network's N total nodes such that $k < n < N$.

2. Multicast groups have been created and initialized, with group leaders identified and configured.

3. Nodes in the network are made aware of the identities of the capability group controller nodes $\{CGC_1, CGC_2, ..., CGC_n\}$, of which k partial signatures must be obtained for proper capability operations.

4.2.2 Multicast Capability Group Controllers

Using a threshold-based approach with multiple distributed authorities is not only desirable but effectively necessary in MANETs. Unlike unicast EPIC, in multicast environments, the necessity of either distributed or centralized authorities mandates the effective security, both localized and remote, of the nodes issuing capabilities. Being distributed among multiple entities, high availability and fault tolerance are two inherent aspects of threshold cryptography mechanisms.

We can identify the requirements of the distributed capability group controller as follows.

1. A $\{k, n\}$ threshold cryptography mechanism exists and can generate a known signature for some message or quantity M provided at least k of the n participating nodes have contributed a valid share.

2. A given node S constructs a non-permanent proposed capability C_S according to acceptable global parameters and forwards this to a well-known subset of capability-issuing nodes.

3. Authority nodes $\{k_0, k_1, ..., k_n\}$ validate the proposed capability C_S, identifying the sender and appropriate parameters, and issue a partial signature to the sender S.

4. When at least k valid partial signatures are received, S can use the reconstructed quantity as an authorized capability BC to send capability-enabled messages to the given multicast group using the universally verifiable signature attached to the capability.

5. A reconstructed capability BC must be verifiable through the use of a well-known public key corresponding to the issuing authority.

4.2.3 EPIC-M Protocol Design

EPIC-M Notation

We use the following notation to describe the EPIC-M protocol:

R, S	R and S are communicating principals: S, the source, wishes to communicate with the destination R
N	An intermediate node of the route
ID_X	The identifier for a given node X
K_X^{-1}	Node X's private key
K_X	Node X's public key, derivable from X's identifier ID_X
$\langle M \rangle_{K_X^{-1}}$	Field or message M signed with node X's private key
$\langle M \rangle_{CGC_{i,k}{}_X^{-1}}$	Field or message M signed with node $CGC_{i,k}$'s partial key share
$E\{M\}_{K_X}$	Field or message M encrypted with node X's public key
f_h	One-way hash function used for generating a hash chain
C_S	The sender-generated proposed capability
BC	The signed base capability, constructed from at least k partial signatures
CAP_i	Capability with sequence number i
T_0	Initial capability request time
T_{CV}	Validity period for a given capability chain
G_i	The identifier for a multicast group i
CGC_i	The $\{k, n\}$ threshold CGC for a given multicast group i
$CGC_{i,k}$	The ID for a given node k as a member of CGC_i
$CGC_{i,k}{}^S$	A partial key share for $CGC_{i,k}$

The capability request process is described as follows:

EPIC-M Capability Request

- *S: Compute* CH_n *and* CH_0
- *S: Compute capability* $C_S = (T_0, T_{CV}, ID_S, G_i, CH_0)$
- *S: Compute* $M_1 = (C_S, RFC)$

- $S \rightarrow CGC_{i,k}$: $\langle M_1 \rangle_{K_S^{-1}}$

When a node S wishes to obtain a capability, it creates a proposed capability, initially unsigned, for use in future sending to a multicast group G_i. Since the sender is responsible for generating the hash chain, only the anchor CH_0 is included, but the chain validity period T_{CV} must also be used to achieve effective rate limiting. The proposed unsigned capability indicates the establishment time, the validity period, the sender's ID, the hash chain anchor, and the multicast group identifier. While unsigned, it is only valid as part of a request. Encrypting each message with the public keys of the n CGC nodes ensures that other nodes cannot attempt to manipulate the process or generate phony partial signatures.

However, since k unique partial key shares are required to reconstruct a valid signed capability, at least k unicast requests must be sent. Depending on the network's parameters and effective packet delivery ratio, some buffer can be included and m requests can be sent such that $k \leq m \leq n$. This represents a tradeoff between an increase in overhead and an increase in performance, noting that a more conservative time-based minimum approach can be adopted such that only the necessary number of requests are sent.

Intermediate nodes in the network do not need to perform any validation prior to forwarding requests. It follows that some network attacks could be carried out by attempting to flood or redirect request packets, but this is not unique to EPIC or EPIC-M, either unicast or multicast, and as such denial-of-capability (DoC) prevention methods are a requirement for EPIC-M.

EPIC-M Capability Response

- *$CGC_{i,k}$: Check capability C_S for validity* (T_{CV}, G_i, ID_s)
- *$CGC_{i,k}$: Compute response* $M_2 = \langle M_1 \rangle_{CGC_{i,k}X^{-1}}$
- $CGC_{i,k} \rightarrow S$: $\langle M_2 \rangle_{K_S}$
- *S reaches the threshold k and creates a signed capability BC:*

- S: $k * \langle C_S \rangle_{CGC_{i,k}{}_X^{-1}}$: $C_S \rightarrow BC$

When the n member nodes of the addressed capability group controller receive an authenticated capability request, they first check to ensure the proposed capability C_S is valid. This includes verifying the sender's signature, the target group, and the proposed validity period. If the criteria are met, a CGC node $CGC_{i,k}$ will generate a partial signature of the proposed capability and return it to the sender encrypted with the sender's public key K_S. The returned partial signatures are encrypted with the sender's public key, ensuring only the original sender will be able to collect partial signatures and reconstruct a valid capability. When the sender S has collected at least k partial signatures from the capability group controller, the proposed capability C_S can be used to construct an authorized capability BC. With both unicast and multicast, EPIC-M conducts the capability request-response operation in a point-to-point fashion.

EPIC-M Protocol Operation: Authorized Packets

- *S: Compute Individual Capabilities $CAP_i = \{BC, CH_i, i\}$*
- *$S \rightarrow G_i$: Data Packet | CAP_i*
- *N_i:* ***No prior knowledge of*** *BC*
  - *The sender includes the signed capability BC with the first packet of the flow ($i = 0$). Otherwise, the previous node on the route, having detected a route change (a different next hop), will include the signed BC with the data packet. Verify authenticity of BC with well-known group public key K_G and check that it has not expired. Verify that $CH_0 = f_h^i(CH_i)$ using CH_0 included in BC. Check that target multicast group G_i in BC matches current group address. Cache BC and CH_i along with the next hop.*
- *N_i:* ***Cached*** *BC*

– *Verify $CH_{i-1} = f_h(CH_i)$. Check that capability has not expired and well-known public key and group parameters have not changed. If packet is outdated or a duplicate, drop the packet. If the next hop has changed, include cached BC in the data packet.*

Multicast capability verification expands upon unicast capability verification slightly. With the knowledge that multicast group capability group controllers and well-known public keys associated with a group can change, it becomes necessary to ensure that a capability is still valid beyond using parameters beyond simply the validity period. For our purposes, we assume that public key redistribution or threshold re-keying is beyond the scope of this dissertation.

Nodes must maintain state information to effectively process capabilities. For a given node n, it must maintain state for a maximum number of $(a * b)$ separate flows, where a denotes the number of multicast groups that the node routes messages for and b represents the number of authorized senders in that multicast group that route traffic through n.

EPIC-M Multicast Capability Maintenance

As capabilities in EPIC-M are issued by an entity other than the receiver, member nodes responsible for routing multicast messages must be aware of and cache well-known keys. Note that nodes do not need to be aware of the identities of the capability group controller nodes unless they want to send data to a multicast group.

With respect to EPIC-M, the multicast group reorganization that occurs at a protocol level following a group leader change does not immediately invalidate existing capabilities issued by that authority. Other changes - tree pruning and membership change, for example - also do not necessarily affect existing capabilities. The increased cryptographic demands placed on both sender nodes and CGC nodes by the multicast model necessitate an approach that minimizes renegotiation.

EPIC-M capabilities are not renewed automatically regardless of the method of expiration. Multicast maintenance messages, when received, will indicate implicitly to authorized

senders that a new capability request is required and must be issued to the updated group leader. This does not require any additional traffic or control messages (no explicit error control notifications are required). However, there are exceptions, such as the case where a CGC group is considered compromised. This means that capabilities signed by that $\{k, n\}$ threshold group are no longer valid. Other maintenance requirements are identical to unicast EPIC in that capabilities that are otherwise exhausted by repeated use or have reached temporal expiration must be renewed by the standard capability request process.

4.3 Multicast Security: EPIC-M

In this section, we evaluate the security of the EPIC-M protocol with respect to various attacks. We use the defined notation from Section 4.2.3, while the protocol is evaluated under the following assumptions:

1. The digital signature scheme and its underlying identity-based cryptosystem are secure.

2. The one-way hash function used in capability generation and verification is secure.

3. A node's private key is secure, preventing one node from masquerading as another.

4. Senders and receivers are not compromised at the time of capability establishment.

As EPIC-M uses the same underlying mechanism as EPIC and RAC, the security analysis with regards to forgery, interception, replay attacks, and data modification as detailed in Section 3.5 apply. Similarly, the structure of the EPIC-M protocol does not alter the security characteristics with regards to protecting capability establishment. With EPIC-M, as with both EPIC and RAC, malicious nodes would need to be able to both successfully forge a capability hash value and guess the bits for each subsequent hop in the route. However, as we consider the hash function to be secure, this does not add any security benefit.

4.3.1 Threshold Cryptography and Capability Group Controllers

With EPIC-M, we use $\{k, n\}$ threshold cryptography to request and establish capabilities. Threshold cryptography allows for both increased availability by increasing the number of nodes available to sign capabilities (n) as well as increased security by requiring a greater number of nodes (k) to sign a proposed capability. The choice of both k and n in this case represent tradeoffs between security and availability. As k increases, the security of the mechanism increases; similarly, as n increases, the availability of the mechanism increases. The use of threshold cryptography mechanisms as capability group controllers allows for capability establishment to continue even when individual nodes are unavailable (providing resilience) or compromised (providing security).

The selection of k and n is ultimately dependent on the security requirements of the network. Maximizing reliability and availability means maximizing the difference between k and n, either by decreasing the value of k or increasing the value of n, while tolerating compromise of the group controller nodes means increasing the value of n. In general, for a $\{k, n\}$ threshold cryptographic mechanism, the following hold true:

- $(n - k)$ node failures are tolerated.
- $(k - 1)$ node compromises are tolerated.

4.3.2 Flooding Attacks / Packet Injection

Unicast EPIC capabilities are potentially subject to remote injection attacks whereby a malicious node, close in spatial proximity to an established and authorized route carrying capability-enabled packets, could forward those packets to a remote location along a high-speed covert link with the goal of flooding the network with authorized packets. Without packets being tied to a specific route, they would be considered authorized and forwarded to the specified destination. RAC introduced dynamic route reconfiguration and authorization, effectively limiting the scope of such an attack by requiring that a forwarding node either be on the current active route or have recently forwarded a request to be included on the

authorized route. This allows for some route changes due to mobility, node membership, or other factors without requiring a wait time for capability renegotiation while simultaneously mitigating the effects of deliberate remote injection.

In EPIC-M, capability-related forwarding decisions are not made until the packet is routed and otherwise destined for sending on an outgoing interface. Remote injection of an authorized packet has different effects depending on the multicast routing protocol, but the attack does not succeed. Nodes who obtain capabilities can attempt to inject packets at remote locations, but two factors prevent this:

- *Packets injected in the existing tree or mesh will not be forwarded unless the node recognizes the receiver and would otherwise forward the packet according to the rules of the multicast routing protocol.*
- *Packets injected will be rejected if they are duplicates. Nodes in the multicast tree or mesh structure will only forward packets that meet the requirements of the multicast routing structure, so no duplicate packets will be routed (a packet, malicious or otherwise, will only be routed and forwarded if it is otherwise part of the multicast routing structure.)*

In both tree-based multicast and mesh-based multicast, certain nodes are identified as forwarding nodes. For tree-based, this includes all non-leaf nodes, while for mesh it simply includes a subset of the group nodes such that all group members are reachable. In both cases, the routing protocol prevents packet injection attacks. A node not responsible for forwarding will not forward a packet regardless of capability status. A node that is part of a forwarding group (either tree or mesh) will forward that packet, but only once, so either the injected packet or the original packet will be rejected as a duplicate and discarded. Without the ability to modify messages, malicious nodes cannot force legitimate multicast forwarding nodes to route any packets they otherwise would not. However, in both cases, the only time a packet is actually forwarded is when the first node proximal to the remote injection site is otherwise a participant in the multicast group. This effectively nullifies

the utility of this attack, as a node in the multicast group would be receiving that packet regardless.

4.4 Performance Results

4.4.1 Test Parameters & Metrics

We evaluate EPIC-M with respect to three key factors. They are as follows:

1. The average capability establishment time based on a $\{k, n\}$ threshold mechanism
2. The performance of the multicast application
3. The associated capability overhead of the multicast application

To analyze these properties, we conduct a series of operations while varying the routing protocol mobility model, node movement speed, and transport-layer application. The details are listed in Table 4.1.

Table 4.1: Multicast Simulation Parameters

Parameter	**Values**
Unicast Routing Protocol	AODV
Multicast Routing Protocols	MAODV, OLSR, PIM-SM
Mobility Models	Manhattan, Random Waypoint
Number of Nodes	50
Simulation Area	2.25 km^2
Node Movement Speed	$\{12, 14\}$ m/s : Manhattan $\{10, 40\}$ m/s : Random Waypoint
$\{k, n\}$ Threshold Members	10
Multicast Application	MCBR
Simulation Time	1800 s
Number of Runs	20

This comparison lets us quantify the effectiveness of EPIC-M as an efficient capability mechanism by establishing a baseline for performance and efficiency with no capability

operations (an unsecured network) and then comparing performance and efficiency with capability establishment and verification mechanisms in place.

4.4.2 Mobility Models

The Manhattan mobility model is designed to simulate vehicle and pedestrian traffic in a city setting. Nodes move along a restricted grid-like pattern at randomly determined speeds with a fixed acceleration and either turn only at the intersection of horizontal and vertical grid lines or continue straight. Manhattan mobility models have the advantage that movement patterns do not intersect in impractical or counterintuitive ways.

The random waypoint (RWP) model is perhaps the most basic mobility model. Nodes start at a given location on the map randomly determined according to a uniform distribution and then simply move to a randomly determined location at a given speed. A key advantage of the RWP model is that the speed is variable between upper and lower bounds, allowing us to simulate both low-speed and high-speed simultaneous traffic. The disadvantage is that the movement pattern is not necessarily realistic as the roads traveled by the nodes are dynamically generated and may intersect or overlap in unnatural ways. Also, in contrast to Manhattan, a node's given position p_0 at some time t_0 does not necessarily have significant influence on its subsequent position p_1 at time t_1 beyond establishing an upper bound on the possible locations based on the maximum movement speed. However, despite these limitations, its simplicity and availability make it an attractive option for simulation.

4.4.3 Multicast Routing Protocols

The main difference in the simulations aside from the presence or absence of the capability mechanisms is with the multicast routing protocol itself. Previous research has shown that the choice of multicast protocol can affect performance substantially with respect to end-to-end delay, packet delivery ratio, and throughput.

1. **MAODV**: The multicast version of AODV, MAODV is a tree-based on-demand multicast routing protocol. A self-organizing protocol, MAODV creates bidirectional

multicast routing trees with a designated leader for each group. Trees are maintained as long as group members remain reachable; disconnected trees can form separate independent multicast trees. When connectivity is reestablished, the original tree can be restored. Each group in an MAODV multicast network is maintained separately, providing a guarantee that the most current routing information available is used.

2. **ODMRP**: The On-Demand Multicast Routing Protocol is a mesh-based multicast routing protocol. This creates multicast meshes instead of trees, potentially asymmetric, and traffic from a given source to a given receiver is not guaranteed to take the same route (nor is the reverse traffic). To reduce overhead, ODMRP utilizes forwarding groups within a multicast group, so only certain nodes are designated to propagate multicast network traffic via localized broadcast.

3. **PIM-SM**: Protocol Independent Multicast - Sparse Mode is a tree-based multicsat routing protocol. Unlike MAODV, trees are unidirectional. It is called protocol independent because it does not have any dependence on any particular underlying unicast routing protocol. PIM-SM is considered more scalable as it builds trees from a given rendezvous point based on explicit membership. In contrast, dense multicast methods generally flood messages and prune areas of the network with no receivers.

EPIC-M Control Overhead Information

We define the following packet control information for all of our EPIC-M simulated models.

To evaluate the performance of EPIC-M, we use the same computation delays as EPIC.

4.4.4 Threshold Capability Establishment

With the use of threshold cryptography to establish multicast capabilities, we expect several tradeoffs to occur:

1. The larger the difference between k and n, the more reliable capability establishment will be.

Table 4.2: Multicast Capability Request: EPIC-M (64 Bytes)

Data Field	**Bit Width**
RFC	4
Control	4
Request Timestamp T_0	24
Sender ID ID_S	32
Multicast Group ID ID_G	32
Capability Validity Period T_{CV}	32
Capability Hash Chain Anchor CH_0	256
Sender Signature S_S	128

Table 4.3: Multicast Capability Response: EPIC-M (64 Bytes)

Data Field	**Bit Width**
RFC	4
Control	4
Sender ID ID_S	32
Multicast Group ID ID_G	32
Capability Validity Period T_{CV}	32
Capability Hash Chain Anchor CH_0	256
Sender Signature S_S	128
Capability Group Controller i Partial Signature S^i_{CGC}	128

Table 4.4: Multicast Capability Data Packets: EPIC-M (40 Bytes)

Data Field	**Bit Width**
Flow ID	16
Sequence Number	16
Multicast Group ID ID_G	32
Current Capability Hash Value CH_i	256

2. For a given value of k, as the value of n increases, capability establishment becomes more reliable, though not necessarily less time-consuming.

Table 4.5: Computation Overhead: EPIC-M

Operation	**Computation Time**
Hash Function	0.5 ms
Digital Signature Generation	20 ms
Digital Signature Verification	60 ms

3. For a given value of n, as the value of k decreases, capability establishment becomes both more reliable and less time-consuming.

These are intuitive results considering that in MANETs, node mobility and the unreliable nature of network communication means that reaching any given subset of nodes is not guaranteed.

Simulation results show that as expected, there is no statistically significant difference between the routing protocols or mobility models. This is because the movement range and node speed is within a close enough range that differences do not reach the level of statistical significance. It is interesting to note that in all cases, ODMRP does trend towards slightly lower capability establishment times, especially at higher thresholds. This is not necessarily an intuitive result as capability establishment is based on the unicast routing protocol, AODV, which is constant across each multicast protocol. However, ODMRP is a more efficient protocol than both MAODV and PIM, so we ascribe the advantages to the reduced multicast overhead.

The distribution of establishment times is exponential rather than linear. This is attributed to the fact that all threshold authority members are contacted at once, not sequentially, and it is increasingly likely that one or more nodes are temporarily unreachable, the route to that node is not stable, or some other complicating factor exists. This affects the selection of both k and n in practice. The average capability establishment times are shown visually in Figures 4.1, 4.2, 4.3, and 4.4.

The rate of increase rises sharply after k reaches 5, and we can see the tradeoff between increased security (requiring a greater threshold be reached) and decreased performance

as capability establishment times increase substantially. We choose a value of $k = 3$ in our simulations, though capability establishment is not necessarily done on-demand but rather some arbitrary time prior to multicast communication. Thus, since these times are not necessarily reflected in the actual capability-enabled multicast performance, we present them here with the caveat that these delays are required prior to authorized multicast communication. However, as the CBR sessions are long-running (10 minutes) the capability establishment delay is unlikely to have any noticeable effect on performance.

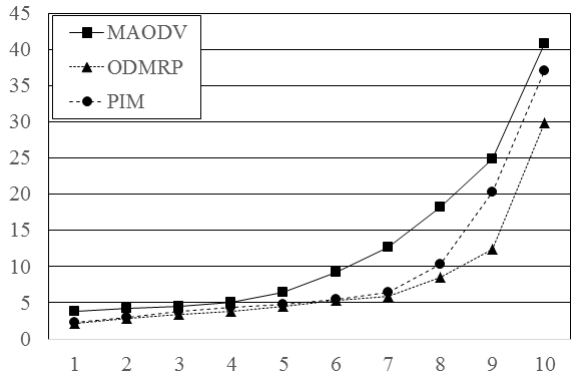

Figure 4.1: $k = \{x, 10\}$ Threshold Capability Establishment Times: Manhattan

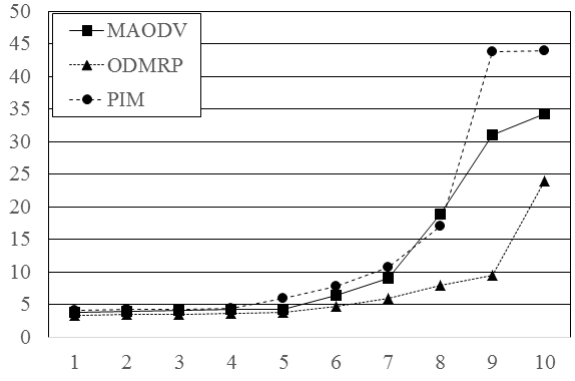

Figure 4.2: $k = \{x, 10\}$ Threshold Capability Establishment Times: RWP

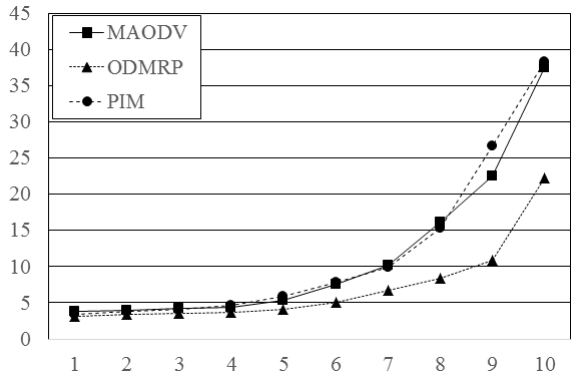

Figure 4.3: $k = \{x, 10\}$ Threshold Capability Establishment Times: Overall Average by Protocol

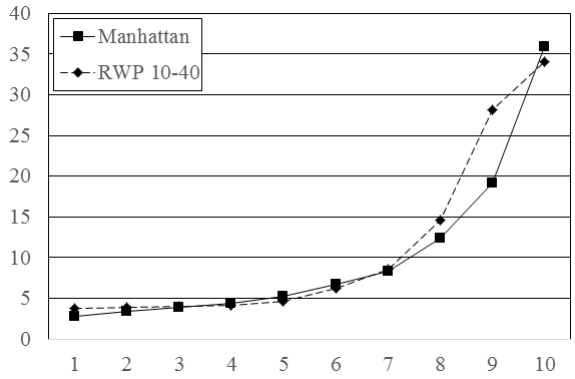

Figure 4.4: $k = \{x, 10\}$ Threshold Capability Establishment Times: Overall Average by Mobility Model

4.4.5 MCBR Performance & Efficiency

We evaluate the performance of EPIC-M using the multicast MCBR application, which sends fixed-rate to multiple group subscribers. In our simulations, ten group members join solely as receivers, and one joins as the primary sender. MCBR is a suitable approximation for practical multicast applications, including real-time or near-real-time compressed video, audio, and data that might be sent in actual multicast environments. Group membership

is defined prior to simulation, but both multicast group operations (control messages) and maintenance multicast capabilities both occur during the actual simulation.

Figures 4.5 and 4.6 illustrate the average multicast CBR throughput at each of the ten receiver nodes. As expected, networks that do not require capability maintenance and verification provide slightly higher performance. However, due to the separate nature of the capability establishment (completely independent of multicast operations), the average actual application-level performance penalty is only 3.5% for Manhattan and 4.5% for RWP. Results show that all of the routing protocols perform well relative to their base performance. For Manhattan, MAODV and ODMRP are the highest-performing while PIM is significantly lower; this is consistent with the established performance of the routing protocols. For RWP, however, the generally lower end-to-end delay associated with PIM-SM leads to increased performance with the higher movement speeds and less predictable movement patterns.

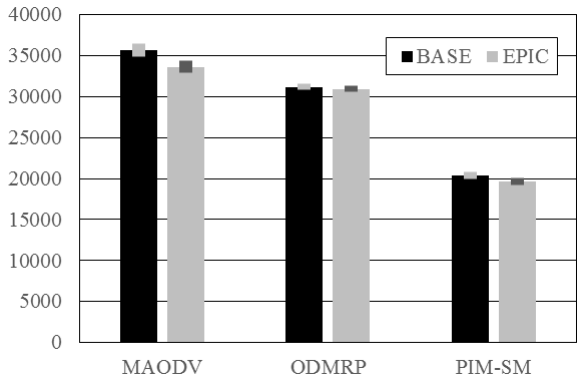

Figure 4.5: Multicast Performance (MCBR Receiver Throughput): Manhattan

Figures 4.7 and 4.8 illustrate the efficiency of each scenario, defined as the difference in overhead and control packets between the base scenario (no capability-enabled traffic) and EPIC-M. Results show that the efficiency penalty averages only 1.9% for Manhattan and 2.1% for RWP, indicating that EPIC-M is a lightweight and efficient method for capability-enabled communication in multicast networks. MAODV and PIM both incur

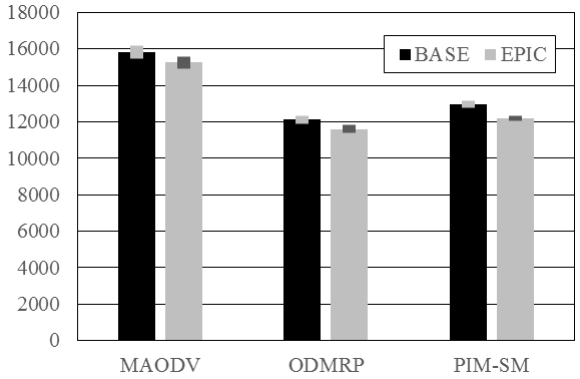

Figure 4.6: Multicast Performance (MCBR Receiver Throughput): Random Waypoint

the most overhead overall as well as the greatest penalty when expanding to capability-enabled traffic, both for Manhattan and RWP scenarios. Thus, we see that the potential performance benefits of MAODV are offset somewhat by increased overhead. The primary source of control overhead associated with EPIC-M is from new nodes in both tree and mesh architectures requesting existing capability information from neighbors; we expect that this would scale as mobility increases (and as such the rate of turnover in the multicast forwarding architecture increases).

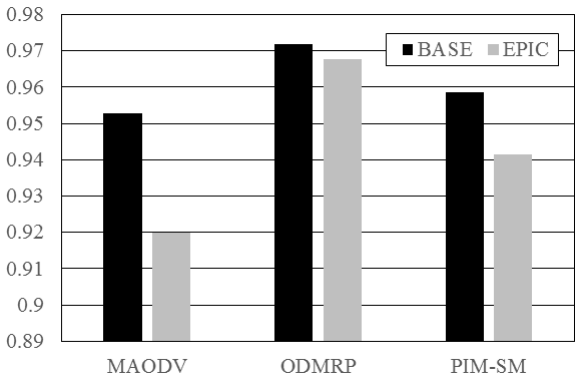

Figure 4.7: Multicast Efficiency (EPIC-M Overhead): Manhattan

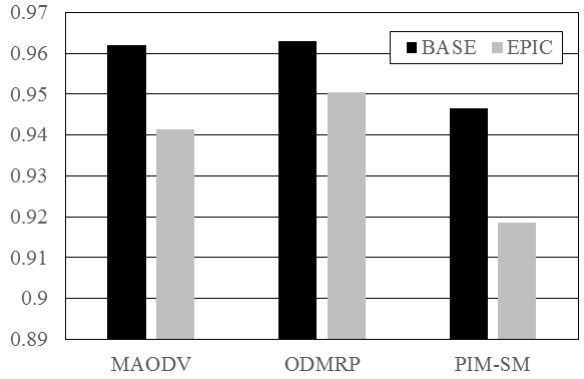

Figure 4.8: Multicast Efficiency (EPIC-M Overhead): Random Waypoint

4.5 Summary

In this section, we have extended EPIC (Efficient Path-Independent Capabilities) to multicast environments. We have presented EPIC-M (Multicast), which uses both identity-based and threshold cryptography to provide secure capability generation and verification in multicast networks. EPIC-M accomplishes this by decoupling operation of the multicast protocol from the operation of the capability protocol. This allows for substantial flexibility in differentiating between send and receive capabilities as well as allowing for management of multiple multicast groups independently. EPIC-M provides the ability for any node in the network to verify a multicast capability; the only prerequisite is that each node be aware of the authority nodes responsible for capability issuance and management.

Simulation results have shown that for a $\{k, n\}$ threshold mechanism, with n set at 10 (20% of the available nodes in the networks), capability establishment time grows slowly from approximately $1 \leq k \leq 5$, with capability establishment times less than $5s$. We also show that impacts on performance and efficiency are relatively minimal with respect to no capability mechanism, with performance decreasing by an average of 4.0% and efficiency decreasing by an average of 2.0%.

Chapter 5: Future Directions and Conclusions

5.1 Summary

In this dissertation, we have presented a contribution to the field of mobile ad hoc network security in the form of three network security mechanisms. Two are unicast security mechanisms. The first is EPIC (Efficient Path-Independent Capabilities), which represents an improvement in efficiency and performance over previous route-dependent existing unicast capability methods. The second mechanism is an enhanced security version of the mechanism, RAC (Route-Adaptive Capabilities), a capability mechanism which dynamically reestablishes capabilities along new routes. This represents an efficient and reasonably well-performing attack-resistant capability mechanism ideal for use in cases where malicious nodes are present or the underlying network in potentially unsecured. EPIC and RAC mechanisms combine the key principles of reverse-disclosure hash chains, which allow two communicating entities to efficiently maintain secure communications over multiple time periods on a unique per-packet basis, and identity-based cryptography, which allows digital signatures and authentication methods to depend on a node's unique identity rather than a shared public key. Simulation results for both EPIC and RAC indicate a statistically significant increase in performance and a significant reduction in routing-associated and capability-associated overhead. Security analysis and simulation also show that both EPIC and RAC mechanisms provide excellent resistance to DoS and DDoS attacks in networks relatively dominated by malicious nodes, providing a quality-of-service level that is highly resistant to even substantial network-layer and application-layer malicious activity.

The third contribution is the multicast security mechanism, EPIC-M (Multicast), wh ich provides a unique method that utilizes both identity-based and threshold cryptography to provide secure network capability generation and verification in multicast networks.

EPIC-M accomplishes this by fully decoupling operation of the multicast protocol from the operation of the capability protocol. EPIC-M provides the ability for any node in the network to verify a multicast capability. As with EPIC, EPIC-M operates efficiently in well-behaved networks, incurring only a minor overhead penalty with respect to unsecured network operations. EPIC-M is also resistant to typical attacks carried out in multicast networks.

5.2 Future Work

5.2.1 EPIC and RAC: Unicast Capabilities

With results being encouraging to this point, future work will be focused on three primary areas. The first will be the exploration of using hop counts to make forwarding decisions, potentially allowing for different routes provided the route length is within a given distance of the original (established) capability route. The second will be to evaluate the possibility of dynamic adaptation, potentially leveraging anomaly detection and intrusion detection methods to change the network's forwarding restrictions when attacks are detected. This would ideally let the network use EPIC in networks where the level of malicious activity is low while switching to a more restrictive EPIC mechanism, either the aforementioned hop count-limited method or RAC, when the level of malicious activity increases. The third will be the implementation, deployment, and evaluation of EPIC, RAC, and EPIC-M in real-world situations, including complex attack scenarios (such as subsets of colluding nodes). As a general goal, we plan to adopt aspects of EPIC, RAC, and EPIC-M into a high-level deny-by-default MANET architecture suitable for most mobile networks.

5.2.2 EPIC-M: Multicast Capabilities

Future work includes the simulation of attackers in multicast networks, the dynamic maintenance of multicast capabilities during normal operations, and finally the real-world implementation and deployment of EPIC-M. The first point is important as capabilities are

designed to mitigate or prevent attackers from impacting the network. The second point is important as well as we have already shown that capability establishment is not necessarily trivial from a temporal perspective and the impact on performance of dynamic capability maintenance is nontrivial. We can study these effects on a new-node basis (a node joins an active group, wishing to send immediately, but must first establish a capability), an existing-node basis (a node possessing an active capability must renew it because it has expired or otherwise invalid), or a group-wide basis (the threshold authority membership for a particular group has changed, so all capabilities must be renewed).

Bibliography

Bibliography

[1] D. Djenouri, L. Khelladi, and A. Badache, "A Survey of Security Issues in Mobile Ad Hoc and Sensor Networks," *IEEE Journal of Communications Surveys*, vol. 7, no. 4, Q4 2005.

[2] B. Wu, J. Chen, J. Wu, and M. Cardei, "A survey of attacks and countermeasures in mobile ad hoc networks," in *Wireless Network Security*. Springer US, 2007, pp. 103–135.

[3] B. Zhu, Z. Wan, M. Kankanhalli, F. Bao, and R. Deng, "Anonymous Secure Routing in Mobile Ad-hoc Networks," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, November 2004.

[4] P. Papadimitratos and Z. Haas, "Secure Routing for Mobile Ad Hoc Networks," in *Proceedings of the 2002 SCS Communication Networks and Distributed Systems Modeling and Simulation Conference*, January 2002.

[5] R. Stoleru, H. Wu, and H. Chenji, "Secure Neighbor Discovery in Mobile Ad Hoc Networks," in *Proceedings of the 8th IEEE International Conference on Mobile Ad-Hoc and Sensor Systems*, October 2011.

[6] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "Establishing Pairwise Keys for Secure Communication in Ad Hoc Networks," in *Proceedings of the 11th IEEE International Conference on Network Protocols*, November 2003.

[7] ——, "LHAP: A Lightweight Hop-by-Hop Authentication Protocol for Ad-Hoc Networks," in *Proceedings of the 23rd International Conference on Distributed Computing Systems*, 2003.

[8] S. Zhu, S. Setia, and S. Jajodia, "LEAP+: Efficient Security Mechanisms for Large-Scale Distributed Sensor Networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 4, November 2006.

[9] H. Ballani, Y. Chawathe, S. Ratnasamy, T. Roscoe, and S. Shenker, "Off by Default!" in *Proceedings of the 4th Workshop on Hot Topics in Networks*, November 2005.

[10] T. Anderson, T. Roscoe, and D. Wetherall, "Preventing Internet Denial-of-Service with Capabilities," in *Proceedings of the 2nd Workshop on Hot Topics in Networks*, November 2003.

[11] A. Yaar, A. Perrig, and D. Song, "SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks," in *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, May 2004.

[12] A. Stavrou and A. Keromytis, "Countering DoS Attacks with Stateless Multipath Overlays," in *Proceedings of the 12th ACM Conference on Computer and Communications Security*, 2005.

[13] K. Argyraki and D. Cheriton, "Scalable Network-Layer Defense Against Internet Bandwidth-Flooding Attacks," *IEEE/ACM Transactions on Networks*, vol. 17, no. 4, August 2009.

[14] X. Yang, D. Wetherall, and T. Anderson, "A DoS-Limiting Network Architecture," in *Proceedings of the 11th ACM Special Interest Group on Data Communications Conference*, August 2005.

[15] T. Wolf and K. Vasudevan, "A High-Performance Capabilities-Based Network Protocol," in *Proceedings of the 5th IEEE Workshop on Secure Network Protocols*, October 2009.

[16] B. Parno, A. Perrig, and D. Andersen, "SNAPP: Stateless Network-Authenticated Path Pinning," in *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security*, March 2008, pp. 168–178.

[17] M. Alicherry, A. Keromytis, and A. Stavrou, "Deny-by-Default Distributed Security Policy Enforcement," in *Proceedings of the 5th International ICST Conference on Security and Privacy in Communication Networks*, September 2009.

[18] X. Liu, X. Yang, and Y. Xia, "NetFence: Preventing Internet Denial-of-Service with Capabilities," in *Proceedings of the 2010 ACM Conference of the Special Interest Group on Data Communication*, September 2010.

[19] A. Snoeren, C. Partridge, L. Sanchez, C. Jones, F. Tchakountio, B. Schwartz, S. Kent, and W. Strayer, "Single-Packet IP Traceback," *IEEE/ACM Transactions on Networking*, vol. 10, no. 6, December 2002.

[20] J. Naous, M. Walfish, A. Nicolosi, D. Mazieres, M. Miller, and A. Seehra, "Verifying and Enforcing Network Paths with ICING," in *Proceedings of the 7th Conference on Emerging Networking Experiments and Technologies*, December 2011.

[21] T. Kim, C. Basescu, L. Jia, S. Lee, Y. Hu, and A. Perrig, "Lightweight Source Authentication and Path Validation," in *Proceedings of the 2014 Conference of the ACM Special Interest Group on Data Communication*, August 2014.

[22] A. Perrig, R. Canetti, J. Tygar, and D. Song, "The TESLA Broadcast Authentication Protocol," *RSA CryptoBytes*, 2002.

[23] A. Perrig, R. Canetti, D. Song, and J. Tygar, "Efficient and Secure Source Authentication for Multicast," in *Proceedings of the Network and Distributed System Security Symposium*, February 2001.

[24] S. Rafaeli and D. Hutchison, "A Survey of Key Management for Secure Group Communication," *ACM Computing Surveys*, September 2003.

[25] C. Wong, M. Gouda, and S. Lam, "Secure Group Communications using Key Graphs," *IEEE/ACM Transactions on Networking*, February 2000.

[26] T. Kaya, G. Lin, G. Noubir, and A. Yilmaz, "Secure Multicast Groups on Ad Hoc Networks," in *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks*, October 2003.

[27] D. SuganyaDevi and G. Padmavathi, "Secure Multicast Key Distribution for Mobile Ad Hoc Networks," *International Journal of Computer Science and Information Security*, Februrary 2010.

[28] H. Chu, L. Qiao, K. Nahrstedt, H. Wang, and R. Jain, "A Secure Multicast Protocol with Copyright Protection," *ACM SIGCOMM Computer Communications Review*, vol. 32, no. 2, April 2002.

[29] R. Curtmola and C. Nita-Rotaru, "BSMR: Byzantine-Resilient Secure Multicast Routing in Multihop Wireless Networks," *IEEE Transactions on Mobile Computing*, April 2009.

[30] C. Shields and J. Garcia-Luna-Aceves, "KHIP - A Scalable Protocol for Secure Multicast Routing," *ACM SIGCOMM Computer Communications Review*, vol. 29, no. 4, August 1999.

[31] S. Roy, V. Addada, S. Setia, and S. Jajodia, "Securing MAODV: Attacks and Countermeasures," in *2005 Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, September 2005.

[32] R. Shyamala and S. Valli, "Securing Route Discovery in MAODV for Wireless Sensor Networks," *Ubiquitous Computing and Communication*, 2009.

[33] F. He, K. Hao, and H. Ma, "S-MAODV: A Trust Key Computing Based Secure Multicast Ad-Hoc On Demand Vector Routing Protocol," in *Proceedings of the 2010 3rd International Conference on Computer Science and Information Technology*, July 2010.

[34] P. Judge and M. Ammar, "GOTHIC: A Group Access Control Architecture for Secure Multicast and Anycast," in *Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies*, June 2002.

[35] T. Bartczak and P. Zwierzykowski, "Performance Evaluation of Source-Specific Multicast Routing Protocols for IP Networks," in *Proceedings of the 2012 8th International Symposium on Communication Systems and Networks Digital Signal Processing*, July 2012, pp. 1–6.

[36] P. Judge and M. Ammar, "Security Issues and Solutions in Multicast Content Distribution: A Survey," *IEEE Network*, vol. 17, no. 1, January 2003.

[37] M. Alicherry and A. Keromytis, "Securing MANET Multicast using DIPLOMA," in *Proceedings of the 5th International Conference on Advances in Information and Computer Security*, November 2010.

[38] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y. Hu, "Portcullis: Protecting Connection Setup from Denial-of-Capability Attacks," in *Proceedings of the 2007 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, October 2007.

[39] K. Argyraki and D. Cheriton, "Network Capabilities: The Good, the Bad, and the Ugly," in *Proceedings of the 4th Workshop on Hot Topics in Networks*, November 2005.

[40] R. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems," *Communications of the ACM*, vol. 21, no. 2, February 1978.

[41] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," *IEEE Network*, vol. 13, no. 6, pp. 24–30, November 1999.

[42] J. Li, D. Wei, and H. Kou, "Identity-Based and Threshold Key Management in Mobile Ad Hoc Networks," in *2006 International Conference on Wireless Communications, Networking and Mobile Computing*, September 2006.

[43] H. Deng, A. Mukherjee, and D. Agrawal, "Threshold and Identity-Based Key Management and Authentication for Wireless Ad Hoc Networks," in *Proceedings of the 2004 International Conference on Information Technology: Coding and Computing*, April 2004.

[44] S. Yi and R. Kravets, "MOCA: Mobile Certificate Authority for Wireless Ad Hoc Networks," University of Illinois, Tech. Rep., December 2004.

[45] J.Kong, Z. Petros, H. Luo, S. Lu, and L. Zhang, "Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks," in *Proceedings of the Ninth International Conference on Network Protocols*, November 2001.

[46] H. Luo, J. Kong, P. Zerfos, S. Lu, and L. Zhang, "URSA: Ubiquitous and Robust Access Control in Mobile Ad-Hoc Networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, December 2004.

[47] N. Saxena, G. Tsudik, and J. Yi, "Threshold Cryptography in P2P and MANETs: The Case of Access Control," *Computer Networks*, vol. 51, no. 12, August 2007.

[48] M. Alicherry, A. Keromytis, and A. Stavrou, "Evaluating a Collaborative Defense Architecture for MANETs," in *Proceedings of the IEEE Workshop on Collaborative Security Technologies*, December 2009.

[49] M. Alicherry and A. Keromytis, "DIPLOMA: Distributed Policy Enforcement Architecture for MANETs," in *Proceedings of the 4th International Conference on Network and System Security*, September 2010.

[50] J. Baek, J. Newmarch, R. Safavi-Naini, and W. Susilo, "A Survey of Identity-Based Cryptography," in *Proceedings of the 2004 Australian Unix Users Group Annual Conference*, September 2004.

[51] P. Michiardi and R. Molva, "IDHC: ID-Based Hash Chains for Broadcast Authentication in Wireless Networks," Institut Eurecom, Tech. Rep., July 2004.

[52] Y. Hu, A. Perrig, and D. Johnson, "Packet Leashes: A Defense against Wormhole Attacks in Wireless Networks," in *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, April 2003.

[53] T. Hayajneh, P. Krishnamurthy, D. Tipper, and T. Kim, "Detecting Malicious Packet Dropping in the Presence of Collisions and Channel Errors in Wireless Ad Hoc Networks," in *Proceedings of the 2009 IEEE International Conference on Communications*, June 2009, pp. 1–6.

[54] M. Just, E. Kranakis, and T. Wan, "Resisting Malicious Packet Dropping in Wireless Ad Hoc Networks," in *Proceedings of the Second International Conference on Ad-Hoc, Mobile, and Wireless Networks*, October 2003.

[55] R. Nojima and Y. Kadobayashi, "Cryptographically Secure Bloom Filters," *Transactions on Data Privacy*, vol. 2, no. 2, August 2009.

[56] E. Chung, J. Joy, and M. Gerla, "DiscoverFriends: Secure Social Network Communication in Mobile Ad Hoc Networks," in *Proceedings of the 2015 International Wireless Communications and Mobile Computing Conference*, August 2015.

[57] A. Broder and M. Mitzenmacher, "Network Applications of Bloom Filters: A Survey," in *Internet Mathematics*, October 2002.

[58] D. Ma, "Practical Forward Secure Sequential Aggregate Signatures," in *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security*, March 2008.

[59] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps," in *Proceedings of the 22nd International Conference on Theory and Applications of Cryptographic Techniques*, May 2003.

[60] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters, "Sequential Aggregate Signatures and Multisignatures without Random Oracles," in *Proceedings of the 24th Annual International Conference on The Theory and Applications of Cryptographic Techniques*, May 2006.

[61] M. Zhao, S. Smith, and D. Nicol, "Aggregated Path Authentication for Efficient BGP Security," in *Proceedings of the 12th ACM Conference on Computer and Communications Security*, November 2005.

[62] S. Chakrabarti, S. Chandrasekhar, M. Singhal, and K. Calvert, "Authenticating DSR using a Novel Multisignature Scheme based on Cubic LFSR Sequences," in *Proceedings of the 4th European Conference on Security and Privacy in Ad-Hoc and Sensor Networks*, July 2007.

[63] R. Bhaskhar, J. Herranz, and F. Laguillaumie, "Efficient Authentication for Reactive Routing Protocols," in *Proceedings of the 20th International Conference on Advanced Information Networking and Applications*, April 2006.

[64] L. Carrea, A. Vernitski, and M. Reed, "Yes-No Bloom Filter: A Way of Representing Sets with fewer False Positives," *Computing Research Repository*, March 2016.

[65] R. Jain, *The Art of Computer Systems Performance Analysis - Techniques for Experimental Design, Measurement, Simulation, and Modeling.* Wiley, 1991.

[66] X. Zeng, R. Bagrodia, and M. Gerla, "GloMoSim: A Library for Parallel Simulation of Large-Scale Wireless Networks," in *Proceedings of the 12th Workshop on Parallel and Distributed Simulation*, May 1998.

[67] P. Sommer, "Design and Analysis of Realistic Mobility Models for Wireless Mesh Networks," Master's thesis, ETH Zurich, 2007.

[68] R. Baumann, F. Legendre, and P. Sommer, "Generic Mobility Simulation Framework (GMSF)," in *Proceedings of the 1st ACM SIGMOBILE Workshop on Mobility models*, May 2008.

[69] G. Ateniese, G. Bianchi, A. Capossele, and C. Petrioli, "Low-Cost Standard Signatures in Wireless Sensor Networks: A Case for Reviving Pre-computation Techniques," in *Proceedings of the 20th Annual Network and Distributed System Security Symposium*, February 2013.

[70] H. Wang and Q. Li, "Efficient Implementation of Public Key Cryptosystems on Mote Sensors," in *Proceedings of the 8th International Conference on Information and Communications Security*, December 2006, pp. 519–528.

[71] A. Ali, "Comparison and Evaluation of Digital Signature Schemes employed in NDN Network," *International Journal of Embedded Systems and Applications*, vol. 5, no. 2, June 2015.

[72] S. Phoha, T. L. Porta, and C. Griffin, *Sensor Network Operations*. Wiley-IEEE Press, 2006.

[73] A. Huhtonen, "Comparing AODV and OLSR Routing Protocols," Helsinki Institute of Technology, Tech. Rep., 2004.

[74] H. Paul and P. Sarkar, "A Study and Comparison of OLSR, AODV, and ZRP Routing Protocols in Ad Hoc Networks," *International Journal of Research in Engineering and Technology*, vol. 2, no. 8, August 2013.

[75] S. Mohapatraa and P.Kanungob, "Performance Analysis of AODV, DSR, OLSR and DSDV Routing Protocols using NS2 Simulator," in *Proceedings of the 2011 International Conference on Communication Technology and System Design*, December 2011.

[76] A. Aneiba and M. Melad, "Performance Evaluation of AODV, DSR, OLSR, and GRP MANET Routing Protocols using OPNET," *International Journal of Future Computer and Communication*, vol. 5, no. 1, February 2016.

[77] C. Mbarushimana and A. Shahrabi, "Comparative Study of Reactive and Proactive Routing Protocols Performance in Mobile Ad Hoc Networks," in *Proceedings of the 21st International Conference on Advanced Information Networking and Applications Workshops*, May 2007.

[78] S. Ali and A. Ali, "Performance Analysis of AODV, DSR and OLSR in MANET," Master's thesis, Blekinge Institute of Technology, 2009.

[79] J. Singh and R. Mahajan, "Performance Analysis of AODV and OLSR using OPNET," *International Journal of Computer Trends and Technology*, vol. 5, no. 3, November 2013.

[80] K. Kaur, S. Kaur, and V. Singh, "Throughput Analysis of Proactive and Reactive MANET Routing Protocols," *International Journal of Emerging Research in Management and Technology*, vol. 3, no. 3, March 2014.

[81] P. Bai and M. Sundararajan, "Performance Efficiency of OLSR and AODV protocols in MANETs," *Indian Journal of Science and Technology,*, vol. 8, no. 4, July 2015.

[82] T. Kumar, "Performance Evaluation of AODV and OLSR under Mobility," Master's thesis, Rutgers University, 2009.

[83] P. Kuppusamy, K. Thirunavukkarasu, and B. Kalaavathi, "A Study and Comparison of OLSR, AODV and TORA Routing Protocols in Ad Hoc Networks," in *Proceedings of the 2011 3rd International Conference on Electronics Computer Technology*, April 2011.

Curriculum Vitae

Eric Swankoski was born in 1981 in Allentown, Pennsylvania. He attended high school in his hometown of Northampton, Pennsylvania, prior to attending Pennsylvania State University from 1999 to 2004 where he obtained the Bachelor of Science degree in Computer Engineering (2002) and the Master of Science degree in Computer Science and Engineering (2004) with a thesis focused on encryption and security in reconfigurable hardware. After graduation, he moved to Virginia in 2004 to pursue research and engineering opportunities, first at the U.S. Naval Research Laboratory and later at General Dynamics Information Technology. He enrolled at George Mason University in 2006 to pursue the Ph.D. degree in Computer Science under the advice of Dr. Sanjeev Setia. His research interests include location-aware computing, automation, evidence-based heuristics and computation, and information assurance and security.