UNIVERSITY of CALIFORNIA
SANTA CRUZ

SECURING DISTANCE-VECTOR ROUTING PROTOCOLS

A thesis submitted in partial satisfaction of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Bradley R. Smith

June 1997

The thesis of Bradley R. Smith is approved:

Prof. J.J. Garcia-Luna-Aceves, Chair

Prof. Darrell D. E. Long

Prof. Tracy Larrabee

Dean of Graduate Studies

Copyright © by

Bradley R. Smith

1997

Securing Distance-Vector Routing Protocols

Bradley R. Smith

Abstract

The security requirements of distance-vector routing protocols are analyzed, their vulnerabilities identified, and countermeasures to these vulnerabilities are proposed. The innovation presented involves the use of mechanisms from the path-finding class of distance-vector protocols as a solution to the security problems of distance-vector protocols. The result is the first proposal to effectively and efficiently secures distance-vector protocols in constant space.

Acknowledgments

While any academic achievement, almost by definition, owes much credit to others, this is especially true in the case of my Master's degree. I owe special thanks to Darrell Long whose support and encouragement over the years have more than once kept me going on what turned into quite a long road. Similarly, I owe a special thanks to Pat Mantey — an outstanding manager and mentor, he has both inspired and encouraged me in my academic career. I consider the opportunity to work with these two gentlemen to be one of the good fortunes of my life. I owe J.J. Garcia-Luna ongoing thanks as my thesis committee chair, advisor, and mentor in my chosen research area of network protocol security. It has been a pleasure and privilege to work with him, and I look forward to more productive years of collaboration. I owe Tracy Larrabee thanks for her time as a member of my thesis committee. She has provided much valuable feedback — this thesis has benefited significantly from her input.

While all UCSC students owe thanks to UCSC's staff who make their academic careers possible, I am more indebted than most. I owe special thanks to Gary Moro who has been, over the years, a co-worker of the top caliber and a good friend, and to Dennis Artman who has been a reliable source of encouragement and support. I would also like to acknowledge the Baskin Center staff, both technical and administrative, the staff of the Natural Sciences Business Office and of the Graduate Division who have always had words of encouragement. I'd also like to thank Laurie Salatich of the Registrar's Office who has gone beyond the call of duty numerous times to smooth administrative wrinkles when my staff responsibilities have overwhelmed my academic commitments and caused me to "stretch" deadlines to the breaking point.

Most importantly I thank my family: my parents, whose love, support and belief in me are the foundation of all my achievements; my sisters, whose love and confidence in their big brother are some of my greatest treasures; and Alicia Márquez, my better half, whose patience and support made this achievement possible.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Routing protocols support the delivery of packets, in spite of changes in network topology and usage patterns, by dynamically configuring the routing tables maintained at routers in internets. The compromise of this routing function in an internet can lead to the denial of network service, the disclosure or modification of sensitive routing information, the disclosure of network traffic, or the inaccurate accounting of network resource usage. The primary focus of security services in routing protocols is the protection of routing information from threats to the integrity (the intentional corruption of routing data), authenticity (the acceptance of routing information from an unauthorized entity by legitimate routers), and in some cases the confidentiality (of for example sensitive policy information) [19] of routing updates. The specific strategies and mechanisms most effective at securing this information depend on a number of attributes of routing protocols that determine specifically what information is exchanged and which set of principles interact in the progression of a routing computation.

The routing protocols used in most of today's computer networks are based on shortest-path routing algorithms that can be classified as distance-vector or link-state algorithms. In general, shortest path algorithms operate by computing the shortest path spanning tree of a network rooted at the source node. Distance-vector algorithms are based on the Distributed Bellman-Ford (DBF) algorithm [2], in which a node knows the length of the shortest path from each neighbor node to every network destination and uses this information to implicitly compute the shortest-path spanning tree. Specifically, with this information, a node is able to compute the length of the shortest path and the next node on this path to each destination. A node sends update messages to its neighbors, who in turn process the messages and send messages of their own if needed. Each update message contains a vector of one or more entries, each of which specifies, as a minimum, the distance to a given destination. In contrast, link-state algorithms are based on Dijkstra's algorithm [2], where a node must know the entire network topology to compute the shortest path spanning tree. Each node broadcasts update messages, containing the state of each of the node's adjacent links, to every other node in the network.

Current distance-vector based routing protocols contain few, if any, mechanisms to provide for the security of their operation, and those that exist are often incomplete. For example, the security mechanisms currently defined for BGP [16] and RIPv2 [10] protect the transmission of routing messages across local networks; however, they do not provide integrity or authenticity of the routing information itself as it traverses an internet. These mechanisms require trust of neighbors regarding updates describing the full internet, and transitively, similar trust of all routers in an internet. Solutions to this problem have been proposed (for example, that of Murphy [12]) involving the inclusion of full path information and a digital signature of the selected route in the updates. However, such solutions, requiring the validation of nested signatures, have extreme space and time costs that make them unusable in any practical sense. Solutions for securing link-state protocols have been proposed that address both the security of routing message transmission and of the routing information itself [12, 15]. While this class of routing protocols has the advantage of a more straightforward means of securing routing information in a manner that effectively limits the scope of trust, it also involves considerable computation and space overhead that compromise its usability in large-scale internets. Given the evolution of the global Internet to a commercial, production network infrastructure, this state of affairs is clearly unacceptable.

The proposal presented here addresses all of these issues. It addresses distance-vector protocols specifically, providing security to these protocols, which have an inherent advantage in computation and space overhead compared to link-state protocols. The solution presented here validates the full path represented by a routing update, confirming not only the next hop of a route but, also that the next hop is the first hop on a path that reaches the named destination at the reported cost. Furthermore, this validation only requires trust of routers concerning information regarding links incident on them, eliminating the transitive trust required by previous proposals. Lastly, this security is provided in constant space per destination in an internet.

Chapter 2 gives a brief description of the network model assumed in the following. Chapter 3 contains a brief review of cryptography, and the specific technologies used in the following sections to secure distance-vector routing protocols. Chapter 4 analyzes the security of distance-vector algorithms, and identifies their vulnerabilities and the threats to which they are susceptible. Chapter 5 presents the proposed strategies and countermeasures for securing distance-vector routing algorithms. Chapter 6 reviews related work. Chapter 7 presents conclusions.

Chapter 2

Network Model

A network is modeled as an undirected connected graph where nodes in the graph are routers and links are sub-networks (subnets). Each link has two lengths or costs associated with it — one in each direction. A network is defined by an IP address range, and can either be a single link-level network (often called a local/metropolitan/wide area network or LAN/MAN/WAN), or, recursively, another set of networks sharing an IP address space. Such a set of interconnected networks is often referred to as an internet. Each node (router) and link (subnet) in the graph has a unique id. Each link has a cost, which can vary in time but is always positive. The distance between two nodes is the sum of the link costs in the path of least cost, or *shortest path*, between them. Figure 2.1 shows a map of an example internet, and Figure 2.2 shows a schematic of the same network with arrowed lines drawn to show selected routes for use later in the paper. In these drawings lower case letters identify routers, and upper case letters identify subnets.

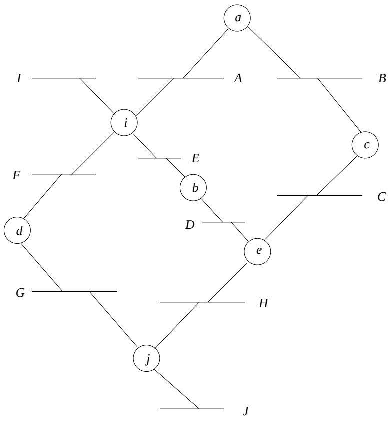

Figure 2.1: Map of Example Internet

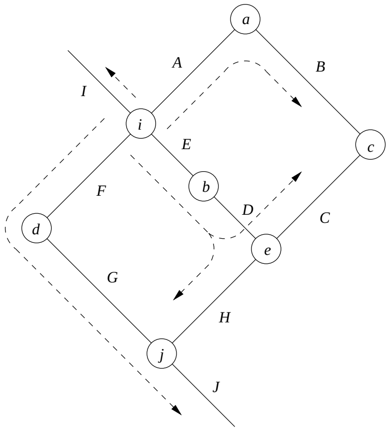

Figure 2.2: Schematic of Example Internet

Chapter 3

Cryptographic Tools

Except where noted, this section is heavily based on Schneier [18], which is a rich source of references for many of the concepts in what follows. The proposed Internet Security Architecture (ISA) [19] provides an architecture for the inclusion of security facilities in the design of protocols to be used in the Internet. Fundamental to the ISA are four concepts:

Vulnerability: A weakness in a system's security that may be exploited by an intruder.

Threat: A potential violation of security. It requires an intruder who has the capability to exploit an existing vulnerability. Threats can be classified into four general categories. *Disclosure* is an event in which an entity gains access to data that the entity is not authorized to receive. *Deception* is an event that results in an authorized entity receiving false data and believing it to be true. *Disruption* is an event that interrupts or prevents the correct operation of system services or functions. And, *usurpation* is an event that results in control of system services or functions by an unauthorized entity.

Security Service: Vulnerabilities and threats are minimized or eliminated through the provision of six security services [14]. *Confidentiality* is the protection of data so it is not made available or disclosed to unauthorized individuals, entities, or processes. *Integrity* is the protection of data so that it is not altered or destroyed in an unauthorized manner. *Authenticity* is the verification of the identity claimed by a system entity. *Access Control* is the protection against unauthorized use of system resources. *Non-Repudiation* is the protection against false repudiation of a communication. *Availability* is the assurance that resources are accessible and usable upon demand by an authorized entity.

Countermeasure: A mechanism or feature that provides a security service. Examples of countermeasures include encryption of network traffic to provide confidentiality, and the use of challenge-response technology for providing authentication of user logins.

This section provides a brief overview of the two fundamental cryptographic technologies used in the following sections to implement countermeasures to routing protocol vulnerabilities: encryption and digital signatures.

3.1 Encryption

A message in its original, usable state is called **plaintext**. A message that has been transformed to conceal its original meaning is called **ciphertext**. The process of transforming plaintext into ciphertext is called **encryption**. The process of transforming ciphertext back to plaintext is called **decryption**. The procedure used for encryption and decryption is called a **cryptographic algorithm**, or a **cipher**. The steps taken to encrypt a message are called **encryption algorithms**, and similarly decryption is performed using a **decryption algorithm**. Modern ciphers use a **key**. The value of a key can range over a set of values called the **keyspace**. Typically the larger the keyspace the more secure the encryption. The value of the key is a parameter to the encryption and decryption functions, along with the plaintext or ciphertext, such that for different keys, the same cipher processing the same input text produces different output text. There are two general forms of such key-based ciphers: symmetric and public-key.

3.1.1 Symmetric Ciphers

Symmetric ciphers are based on algorithms where the encryption and decryption keys can be calculated from each other. In many of these systems the keys are the same. The primary functional differences between symmetric and public-key algorithms are a result of difference in the use and management of the keys in each system. When using symmetric ciphers, by definition, both the sender and receiver of a communication either have the same key, or have enough information to calculate each other's keys. This shared nature of symmetric cipher keys presents a number of problems. One is that, fundamentally, non-repudiation is not possible due to the need to share keys. The other is that the need for confidentiality in key distribution adds significant complexity to the key management mechanisms as compared to public-key systems.

Popular examples of symmetric ciphers include the U.S. Data Encryption Standard (DES), and the International Data Encryption Algorithm (IDEA).

3.1.2 Public-Key Ciphers

Public-key ciphers are based on algorithms where the encryption and decryption keys differ, and cannot be calculated from each other in a reasonable amount of time. The name “public-key” comes from the fact that the encryption key is typically made public. Anybody can encrypt a message, but only the individual with the corresponding decryption key can read the message. Some public key ciphers allow the encryption/decryption process to be performed using either key for encryption, and the other for decryption. As will be explained later, this ability is very useful for the generation of digital signatures. Other ciphers provide a similar capability using a different algorithm for digital signatures, which uses the same keys.

This dual key aspect of public-key ciphers gives rise to some significant differences vis-a-vis symmetric ciphers. When using public-key ciphers, only the owner of a key pair has full encryption/decryption information; the public-key portion of a key pair only gives others access to half the process. Using these ciphers, true non-repudiation is possible. Given adequate private-key management mechanisms, the ability to decrypt a message with the public-key verifies the owner as the source of the message, since only the owner has access to the private-key component. Additionally, public-key ciphers have less demanding requirements of the key exchange mechanism vis-a-vis symmetric ciphers. In a public-key cipher the only key that requires distribution is the public key. Since, by definition, a public key does not require any privacy in its distribution, the key exchange mechanism for public keys only needs to provide integrity and authentication. This significantly simplifies the key distribution mechanism.

Examples of public-key algorithms include the RSA algorithm, named after its three inventors, Ron Rivest, Adi Shamir, and Leonard Alderman, and the ElGamal algorithm (named after its inventor). An example of a public-key algorithm that provides for digital signatures via a separate signature algorithm is the U.S. government's Digital Signature Algorithm.

3.2 Digital Signatures

Digital signatures are the other fundamental building block of cryptographic systems. Their purpose is to duplicate the function of traditional signatures in paper-based systems. In particular, the characteristics of traditional signatures that need to be provided by digital signatures include:

- the signature cannot be repudiated (the signature is bound to the identity of the signer).
- the signature is not re-usable (the signature is bound to the signed document).
- the signed document is unalterable (the document is bound to the signature).

As alluded to earlier, many public-key ciphers have the mathematical properties, or are designed in such a way as to provide effective digital signatures. Using the RSA algorithm, which allows either key to be used for encryption, as a popular example: the signer encrypts the message with their private key, and sends the resulting ciphertext to the recipient; the recipient decrypts the ciphertext with the sender's public key. If the resulting plaintext is a valid message then the requirements of a signature have been met:

- the signature cannot be repudiated since only the sender has the private key that could have created the ciphertext.

- the signature is not re-usable, being a function of the original plaintext and a key, it would not be valid for any other document.

- the signed document is unalterable since modified plaintext would not match the deciphered ciphertext.

While digital signatures implemented only with encryption work, as outlined above, they have a couple of problems: space and speed. Because the signature is an encrypted version of the message, the size of the signature is some significant fraction of the size of the message; this results in a storage requirement of somewhere close to double the size of the original message for storage of the message with its signature. Because the speed of public-key encryption is relatively slow, the requirement of encrypting and decrypting a full message to create and verify a signature in this manner is prohibitive. To address these problems, digital signatures can be implemented with a combination of encryption and a one-way hash function.

A **one-way hash function**, also called a cryptographic checksum, message authentication code (MAC), or message digest, has two mathematical properties that make it useful for implementing digital signatures. First, as a hash function, it is a function that takes an arbitrary size message and converts it to a fixed size string. The output of a hash function provides a kind of fingerprint of the input message. A simple example would be the function returning the byte resulting from the XOR of of all bytes of the input message. Because they are many-to-one functions (the example XOR function maps all possible strings into 256 values), hash functions do not uniquely identify a string, but they do provide a useful validity check. A one-way function is a function that is easy to compute, but hard to reverse. A simple example is x^2, which is easy to compute, while its inverse, $\sqrt{x}$, is difficult.

Combining these two concepts results in a one-way hash function that is a function that computes a fixed length value from an arbitrary size message such that calculating the value from the message is easy, but generating a message that hashes to a given value is very hard. The XOR function mentioned earlier clearly is not a one-way hash function because it is easy to generate strings that hash to a given value. Examples of one-way hash functions include Message Digest 4 (MD4), and Message Digest 5 (MD5), both developed by Ron Rivest, and the Secure Hash Algorithm (SHA), developed by NIST and the NSA for use with the U.S. government's Digital Signature Standard.

Combining one-way hash functions with public-key encryption results in an efficient and effective digital signature. Using a public-key cipher, such as RSA, and a one-way hash function, a digital signature would be implemented as follows: the sender calculates a one-way hash of their message, and encrypts the hash value with their private key; the sender sends the message with the encrypted hash value to the receiver; the receiver calculates the hash value of the received message, and decrypts the received hash value with the sender's public key.

If the calculated and received hash values are the same, then the requirements of a signature have been met:

- The signature cannot be repudiated because only the sender has the private key to encrypt the hash value.

- The signature is not reusable because the one-way hash function is effectively unique for a given document. For the SHA algorithm, which generates a 160 bit hash value, the chances are one in 2^{160} of two documents having the same hash value.

- The signed document is unalterable because even a slight modification of the message would dramatically change the resulting hash value.

Benefits of this signature method are many: the speed is increased dramatically, much less space is needed to store the document with its signature, and the signature may be stored separately from the document. These features provide many benefits — including practical implementations of multiple signatures for a document, and signature archives which prove the existence of a document without requiring the disclosure of its contents.

Chapter 4

Vulnerabilities in Distance-vector Protocols

These concepts are now used to develop a solution for securing distance-vector routing protocols by analyzing the protocol design to identify vulnerabilities and threats, identifying the security services needed to reduce or eliminate the vulnerabilities, and designing the appropriate countermeasures to provide the needed services. The solution presented here only deals with threats to the flow of routing traffic and does not address threats to the flow of data traffic. Attacks are described in terms of different classes of internet nodes, including authorized routers and intruders. Authorized routers are those nodes intended by the authoritative network administrator to participate in the routing dialog and computation, that run correct and bug-free code, and that use correct and bug-free configuration information.

4.1 Intruders

The countermeasures presented here assume that intruders can position themselves at any point in the network through which all traffic of interest flows, and that an intruder has the capability to fabricate, replay, monitor, modify, or delete any of this traffic. Interpreting this description for a routing environment, the following general classes of intruders are identified:

Masquerading routers: A masquerading router is a node that successfully forges an authorized router's identity. This can be accomplished using the IP spoofing [11] or source routing attacks.

Subverted routers: A subverted router is one that is caused to violate the routing protocols or to inappropriately claim authority for network resources. This typically occurs due to bugs in the routing code, mistakes in the configuration information, or by causing a router to load unauthorized software or configuration information. The specifics of how this can occur depend on the design and configuration of the router.

Unauthorized routers: An unauthorized router is a node that is not authorized as a router that manages to circumvent any access control mechanisms in place and participates in the routing dialog and computation. How this is achieved depends on the design and configuration of existing authentication and access control mechanisms.

Subverted links: A subverted link is a channel controlled via access to the physical medium (e.g. network cable-plant, the "air-waves", or the electronics used to access them), or via compromise of the protocols underlying the routing protocol in a manner that allows control of the channel (e.g. the TCP session hijacking attack [7]).

4.2 Threats to Routing Information

There are a number of vulnerabilities that allow a strategically placed intruder to fabricate, modify, replay, or delete routing traffic. With these capabilities an intruder can compromise the network in a number of ways. The modification or fabrication of routing updates allows an intruder to reconfigure the logical routing structure of an internet, potentially resulting in the denial of network service, the disclosure of network traffic, and the inaccurate accounting of network resource usage [4]. The replay or deletion of routing updates blocks the evolution of subsets of the logical routing structure (in response to topological or link cost changes), or resets it to an arbitrary earlier configuration with potential results similar to above. The vulnerabilities these attacks exploit is the lack of access control, authentication, and integrity of routing messages.

In addition, it is relatively easy for an intruder to gain access to routing traffic. The information carried in this traffic describes the next hop to take to reach a destination. This information is available from other sources, such as monitoring authorized traffic to the desired destination for the next hop it uses, and therefore cannot be protected by measures only involving the routing traffic. Additionally, in some distance-vector routing protocols, the routing traffic includes information describing the path taken by traffic to different destinations. In some circumstances this information may be considered confidential. Since the only source of this information is the routing protocol, it should be possible to protect with modifications to the routing protocols only. The vulnerabilities these attacks exploit are the lack of confidentiality of peer links, and the level of trust placed in routers.

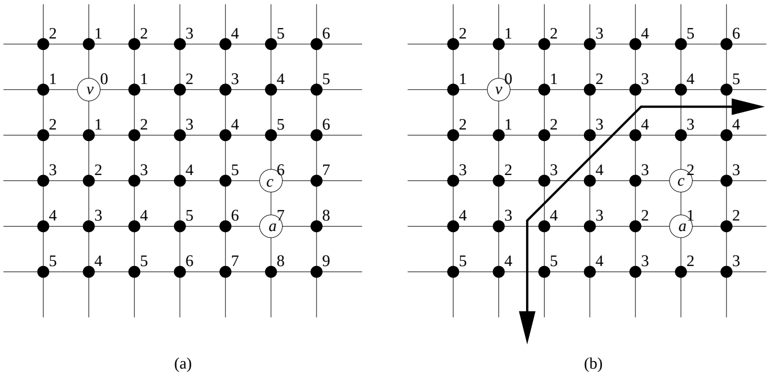

Figure 4.1: Finn's Network Partition Example

4.3 Specific Examples

The network equivalent of a "black-hole" can easily be created by installing a route for a destination with a cost lower than the legitimate route that leads to a non-existent or non-responsive router. All traffic destined for such a network will disappear down this black-hole route, and the affected network will be effectively partitioned from the internet. This is a reasonably common occurrence caused by mis-configured routers, but it can serve a useful purpose in a hostile attack.

Finn describes a subtler version of the black-hole attack [4]. This attack is illustrated in Figure 4.1, where an internet with grid topology and unit link cost is shown with three nodes labeled a, c, and v. Each node is labeled with its distance to network v. At some time after the internet reaches the state shown in Figure 4.1(a), node c sends an update to a indicating a 1 hop route to v. Figure 4.1(b) shows the resulting internet after this update has fully propagated. In the resulting internet all nodes beyond a certain boundary, shown by the dark arrowed line in Figure 4.1(b), have reset their routes to v such that it follows a path that ultimately leads through a to c. Subtle manipulation of the routing topology such as this can allow c to set itself up as a conduit for all traffic between v and a region of the internet.

Rosen [17] documents another vulnerability where corrupted routing packets generated by a node with faulty memory caused the ARPAnet to cease functioning. The problem turned out to be a series of three routing updates generated by the faulty router that caused the routing processes on the receiving routers, in processing the faulty updates, to take over the CPU of these routers causing them to cease any other processing (such as data packet forwarding). Recovery from this problem required rebooting all the routers with modified code. Other possible vulnerabilities include the generation of traffic to or from a forged node in an internet where network fees are based on usage, thus causing inappropriate charges to a node; or snooping routing protocol traffic for a distance-vector protocol that includes path information in the updates (such as BGP), and using it to identify critical resources in an internet to target for an attack. Clearly there are many examples. Some of the examples given here involve mistakes or mis-configurations, however the same vulnerabilities can be exploited for hostile attack — indeed, intentional attacks and accidental mis-configurations are generally indistinguishable from the security countermeasure perspective.

Chapter 5

Distance-vector Protocol Security Countermeasures

In general, the goal of securing distance-vector routing protocols is to provide authenticity, integrity, confidentiality, and access control of routing message transmission. Referring to the previous sections, and speaking in terms of threats and classes of intruders, this goal translates specifically to:

- Prevent the fabrication, modification, and replay of routing messages by all classes of intruder,
- Prevent the deletion of routing messages by subverted links and subverted routers, and
- Prevent the disclosure of routing messages by all classes of intruder.

The countermeasures presented below assume that access control is provided using the same naming and key distribution mechanism used to implement the authentication

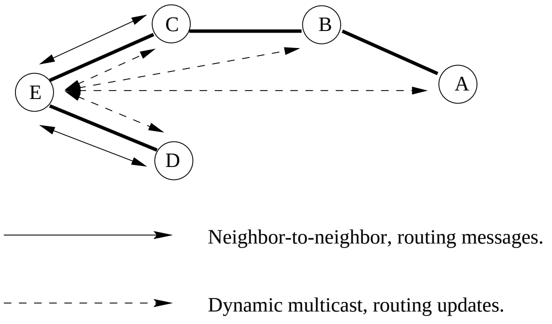
C
B
A
E
D
Neighbor-to-neighbor, routing messages.
Dynamic multicast, routing updates.

Figure 5.1: Classes of Routing Information

mechanism. The remaining access control design issues are orthogonal to the countermeasures presented here, and are not discussed further.

There are two classes of communication occuring in routing protocols (see Figure 5.1): communication between neighboring routers, which is composed of routing updates for destinations the sender has determined are appropriate to forward to the receiver, and communication between a given router and an arbitrary set of remote routers, which is dynamically determined by routing decisions, composed of the fields of routing updates that describe a given destination. Correspondingly, two classes of countermeasures are presented, described in terms of distance-vector algorithms: routing message protection countermeasures and routing update protection countermeasures.

Figure 5.2 shows the additional information used by these countermeasures, and Figures 7.1 and 7.2 specify the primitives required and procedures used to secure a distance-vector routing protocol. A number of assumptions are made in the design of these security mechanisms for distance-vector protocols:

- Intruders have the capabilities described in Section 4.1.

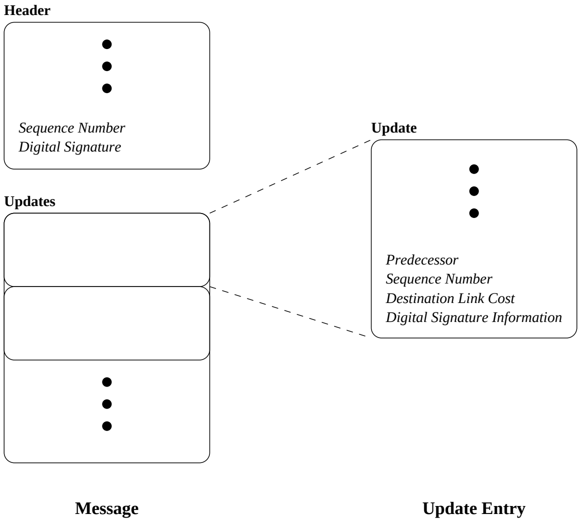

Figure 5.2: Proposed Routing Message Changes

- A router can trust information it receives from other routers only concerning links incident on the remote routers.
- Each router is assigned a public-key pair for use in digitally signing routing messages.
- Key distribution is based on domain names, and domain names can be efficiently and securely determined given an IP address of a host (for example, Secure DNS [3]).

5.1 Routing Message Protection Countermeasures

The following countermeasures are effectively implementing security services not available from lower level transport or network protocols. Specifically, the routing message digital signature and sequence numbers are providing authentication and integrity services of routing messages, which compose the first class of communication described above. If these services were available from network [1] or transport layer protocols, these mechanisms would no longer be needed in the routing protocols.

5.1.1 Routing message sequence number.

A sequence number is included in each routing message. This sequence number is initialized to zero on the initialization of a newly booted router, and is incremented with each message. On detection of a skipped or repeated sequence number a reset of the session is forced by the re-initialization of the routing process. The size of this sequence number is made large enough to minimize the chance of it's cycling back to zero. However, in the event that it does, the session is reset by the re-initialization of the routing process. This protects against the deletion and replay of routing messages.

5.1.2 Routing message digital signature.

Each routing message is digitally signed by the sender. This provides authenticity and integrity (protection from message modification, but not from replay) of routing messages. On detection of corruption, the message is dropped.

5.2 Routing Update Protection Countermeasures

The countermeasures presented in this section protect the second class of communication described above composed of individual routing updates. These countermeasures provide authenticity and integrity of this communication; confidentiality is not necessary as the potential recipients include all authorized routers in an internet.

5.2.1 Add sequence information to updates.

Sequence information is added to each update to protect against the replay of old routing information. This sequence information can be in the form of a sequence number or a timestamp. New sequence information is generated for each route output from the routing selection process. While a number of updates may be generated for each route (one message per neighbor of the originating router), only one sequence number or timestamp is used for all of them. This is necessary as a remote router may receive the same route as a number of updates, each describing the same destination but representing different paths; all of these updates must be considered valid. This implies that updates for a given destination must be considered valid if their sequence information is greater than or equal to the current sequence information. Note that sequence information must be maintained and validated on a per router basis. An invalid update is silently dropped.

In a routing environment sequence information must be valid for the life of a given router id, and therefore has a potentially long life. The primary challenge posed by this requirement of a long life is how to prevent sequence information from cycling. The primary advantage of sequence numbers compared to timestamps is their significantly longer life. Making even aggressive assumptions of the average time between changes in cost of directly connected links (the event that would cause the generation of an update requiring a new sequence number from the local router) a sequence number can be relatively small and still provide reasonable assurance of not cycling. Even assuming local link changes (and the resulting new sequence number) as frequently as every 8 seconds, a four octet sequence number would last for over 1000 years. The main difficulty introduced by sequence numbers is how to maintain them in the context of arbitrary software and hardware failures. Techniques such as those proposed by Perlman [15] could be used; however, if cycling of the sequence number must be supported, the following process can be used:

> Each router maintains an update sequence number database on a per router $< domainname, publickey >$ pair basis. When the cycling of a sequence number approaches, a new public-key pair is generated. The key distribution mechanism and router are updated with the new key pair, and the router's update sequence number is reset to zero. On detection of a change in the public key for an originating router, the receiving router will add an entry to its update sequence number database for the new originating router $< domainname, publickey >$ pair with a sequence number of zero. It will continue to use the old sequence number entry until a sequence number failure occurs where the digital signature validation succeeds using the new entry. At this time the old entry is purged, and the conversion to a new sequence number is complete. Further work is needed on a mechanism to load the database of a newly booted router.

Alternatively, a timestamp could be used. The main benefit of a timestamp would be the ease of administration provided by the well-defined external reference for use in resetting lost state. The life of even a small timestamp, while not as dramatic as for sequence numbers, is still significant; assuming a granularity as small as one second, a four octet timestamp still has a life of over 130 years.

5.2.2 Add predecessor information to updates.

A routing-table update of a distance-vector routing protocol consists of one or more entries, each specifying a destination and a distance to the destination. To verify the integrity and authenticity of a given update entry, the router processing the update must make sure that the distance to the destination reported in the update entry corresponds to a path that is valid and authentic for each of its hops, and that the reported successor is the first hop on this path. By including the information about the second-to-last hop (predecessor) in the path to a destination, the validity and integrity of the entire path from the source router to the destination can by verified iteratively using information reported by the routers directly adjacent to the destination and routers immediately adjacent to each intermediate hop in the path.

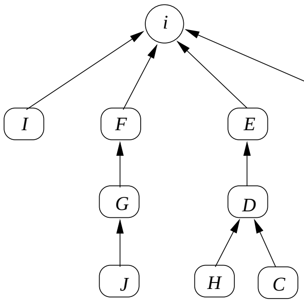

Figure 5.3: Oriented Tree for Example Internet

The use of predecessor information in the reconstruction of the path implied by a route is based on the concept of *oriented trees* [8]. An oriented tree is a directed graph with a distinguished vertex R such that:

- Each vertex $V \neq R$ of the graph has exactly one out bound arc $e[V]$.
- R has no out bound arcs.
- R is the root of the graph (i.e. for all $V \neq R$ there is an oriented path from V to R).

From this definition it is clear there is only one oriented path from each vertex to the root, and therefore there are no oriented cycles. Recalling that the purpose of shortest-path routing algorithms is to compute the shortest-path spanning tree rooted at the source, we see that the predecessor information describing the second-to-last hop of these routes is the arc information, $e[V]$, described above which defines an oriented tree. As an example, the arrows in Figure 2.2 show the paths chosen by router i to reach all destination networks in the internet. Figure 5.3 shows the oriented tree implied by these paths, and the predecessor links which define it. Using this information we can reconstruct the path from a destination to the source and, with the digital signatures described in Section 5.2.4, verify the reported distance. The method used to accomplish this is based on the path-traversal technique of path-finding algorithms.

Path-finding algorithms are distance-vector routing algorithms that maintain predecessor information in addition to the distance and successor information for every destination in the internet. This information is used to compute loop-free paths to all destinations. Using the predecessor information, an implicit path to a destination can be inferred and thus routing loops can be detected. As an example, Table 5.1 shows the routing table maintained at node i for the internet diagramed in Figure 2.2. A routing table is a vector with each entry specifying the destination j, current shortest distance D_j^i, successor s_j^i and the predecessor p_j^i. Let router i want to validate a loop free path to network J. Router i starts the trace at the entry for destination J, and finds that the predecessor to J is G. Subsequently, i walks through the predecessors to the directly connected network F. The sequence of predecessors encountered during such a trace is referred to as the *implicit path*, or the path extracted from the predecessor network information. The working of algorithms of this type is described by Murthy and Garcia-Luna-Aceves [5, 13].

To secure distance-vector protocols the predecessor information defined above is included in each update. Using this information a path traversal for each selected route is then performed, verifying the integrity of the path and the distance reported for the route.

5.2.3 Add destination link cost to updates

To verify the distance advertised in a routing update it must be possible to compute the length of the implied path during the path traversal described in the previous section. For this purpose the cost of the destination link is included in the update by the originating router. This value should be interpreted as the cost for the originating router to reach a node on the destination network. During the path traversal to validate a route these values are totaled, and compared to the advertised distance as part of the validation.

5.2.4 Digitally sign updates.

To ensure the authenticity and integrity of the destination, sequence number, predecessor, and destination link cost update fields described above the originating router includes a digital signature of these fields in the update. Using the authenticated destination, predecessor, and link cost fields a recipient can then verify the distance field as described above. To protect against falsification of destination and predecessor information, and thereby false network topologies, some means of cross-checking the destination and predecessor fields is needed. The following two solutions provide this cross-checking but make different assumptions of the environment, and therefore are appropriate for different routing protocols.

The first solution requires 1) the ability to establish connectivity of a node to a network given only the address and mask of the network and the IP address of the node's interface which is claimed to connect it to the given network, and 2) the existence of a network authority trusted to certify and distribute IP address to node name, and node name to public key mappings (equivalent functionality of secure DNS). With these requirements met, this solution works as follows. The originator of an update includes the IP address of its interfaces to the destination and predecessor networks in the update, and digitally signs these fields in addition to those listed above. A recipient verifies the update by confirming that the given interface addresses belong to the corresponding networks, that both IP addresses map to the same public-key, and that the resulting public-key validates the digital signature. If these tests are successful then, based on the trusted association of interface addresses to networks and interface addresses to the same public-key, the authenticity and integrity of the destination and predecessor networks can be trusted. This solution is appropriate for routing among networks described by IP address ranges, and where routers are directly connected to the networks they route between. This is typical of today's intra-domain routing protocols.

The second solution requires a network authority trusted to certify and distribute router name to connected network associations. Given such an authority, this solutions works as follows. The network authority generates a digital certificate for each router-to-network link in an internet. The originator of a routing update includes the certificates for it's destination and predecessor connections in the update, and signs these fields in addition to those listed above. A recipient verifies the update by verifying the digital certificates for the destination and predecessor networks, and the digital signature using the originator's public-key. If these tests are successful then, based on the trusted certification of the destination and predecessor network connectivity, the authenticity of the destination and predecessor networks can be trusted. This solution is appropriate for environments where either routers are not directly connected to the networks they route between, or where networks are described with arbitrary identifiers (e.g. autonomous system numbers). This is typical of today's inter-domain, policy-based routing protocols.

5.3 Countermeasure Effectiveness

The impact of each countermeasure on the threats described in Section 4 is now analyzed. Table 5.2 summarizes this analysis. In the following it is assumed that the digital signature countermeasure includes facilities for secure authentication and access control. In broad terms, the message protection countermeasures provide protection against all nodes which lack the necessary cryptographic keys, specifically unauthorized routers, masquerading routers, and subverted links. The digital signature of routing messages protects them from fabrication, modification, and disclosure by these classes of intruders. The addition of a sequence number to routing messages protects them from replay or deletion by these intruders.

Similarly, the update protection countermeasures provide protection against the compromise of those nodes that do have the cryptographic keys, specifically subverted routers. The digital signature of each update protects them from fabrication or modification by subverted routers. The addition of sequence number information to each update protects against replay by a subverted router. The addition of the predecessor network to each update provides the information needed to reconstruct and validate the path implied by a route, the successor advertised for a route, and, with the addition of link cost information, the distance advertised for a route.

There were a few vulnerabilities not addressed. Specifically, a subverted router is still able to falsify destination link cost information, delete routing updates, and disclose routing information. If only unit link costs are used (i.e. hop counts) then the path traversal performed using the predecessor information provides complete protection from the first vulnerability. If however, a wider range of link costs are supported, then it is possible for an originating router to “sweeten” a a route by advertising an artificially low cost for it’s link in the path, thereby making the overall costs seem lower than they actually are. Short of using only hop count metrics, there is no way to fully protect against this threat. If a wider range of link costs is needed than hop counts, the best rule of thumb for minimizing exposure to this threat is to keep link costs as low as possible. This leaves the least headroom for intruders to exploit this vulnerability. The source of this vulnerability is the basic assumption stated earlier that routers can be trusted regarding information describing links incident on them. The second and third vulnerabilities are inherent in the requirements of routing protocols to trust routers in their handling of routing updates. It is likely, due to the high degree of connectivity in most operational internets, that the deletion of routing updates will be at worst ineffective in cutting off access to destinations, and at best detectable through the correlation of received routing information. Further research is needed into the possibility of detecting such intrusions.

In addition, it is still possible for any class of intruder to disclose routing information. Due to the multicast nature of these protocols it is not possible to address the threat of the disclosure of routing messages in an efficient manner. To provide this protection would require replacing the routing message digital signature with encryption of the routing message. However, since each message is received by a number of routers this would require sending out a copy of each update encrypted for each recipient router. This requires a significant change in the protocol, and invokes a significant additional cost in both traffic and CPU time for the encryptions. Further research is needed in this area.

5.4 Cost Analysis

The costs for these countermeasures are in the space for new fields, in time for computing the new fields, and in time for performing the path-traversal. Following is a rough summary of these costs. This summary assumes the first solution for digital signatures presented in Section 5.2.4 (that is including the IP addresses for the router's interfaces on both the predecessor and destination networks in the update, and in the update digital signature), IPv4 IP addresses (4 octets), 64 octet digital signatures, a single octet destination link cost, and 4 octet sequence information (timestamps or sequence numbers).

Space per message: Each routing message grows by a 4 octet sequence number and a 64 octet digital signature. This is comparable to security mechanisms currently proposed for some protocols (e.g., RIPv2).

Space per update: Each routing update grows by a 4 octet timestamp, a 1 octet destination link cost, 8 octets for the predecessor network address and mask, 8 octets for the IP addresses of the router's interfaces on the predecessor and destination networks, and a 64 octet digital signature. The total cost per update being 85 octets.

Time per message: A digital signature and sequence number must be computed once for each routing message generated by a router, and verified once per receiving router.

Time per update: The predecessor field of an update differs for each interface of the originating router it is sent over. Therefore, the digital signature of an update must be computed once for each link of the originating router. Conversely, each update digital signature is verified once by each router which selects one or more routes whose implicit paths include the link represented by the update.

Time per destination: Each selection of a new path to a destination requires a path-traversal. The frequency of such changes is dependent on network topology and link change events. However, as has been demonstrated by Garcia-Luna-Aceves and Murthy, efficient path traversal algorithms add minimal overhead [5].

Note that hash chain authentication [6] may be usable for the update digital signature, which would decrease the digital signature space cost to 16 octets, and the time cost to an MD5 calculation. Further research is needed into this option. In addition, the validation actions, which will likely account for a large share of these costs, can be done on a statistical basis to contain time costs.

Destination	Distance	Successor	Predecessor
A	1	–	–
B	2	a	A
C	3	b	D
D	2	b	E
E	1	–	–
F	1	–	–
G	2	d	F
H	3	b	D
I	1	–	–
J	3	d	G

Table 5.1: Routing Table at node i

	Fabrication	Modification	Replay	Deletion	Disclosure
Unauthorized router	A	A	A	A	
Masquerading router	A	A	A	A	
Subverted link	A	A	A	A	
Subverted router	C,D,E*	C,D,E	B,C,D,E		

* Except destination link cost information.

A) Message protection countermeasures
B) Update sequence number
C) Destination link cost
D) Predecessor
E) Update digital signature

Table 5.2: Summary of Countermeasure Effectiveness

Chapter 6

Related Work

Kumar [9] analyzes the security requirements of network routing protocols, and discusses the general measures needed to secure the distance-vector and link-state routing protocol classes. He identifies two sources of attacks: subverted routers, and subverted links. Since attacks by subverted routers are seen as difficult to detect, and of limited value to the intruder, he focuses his attention on securing protocols from attacks by subverted links. For distance-vector protocols this translates to the modification or replay of routing updates. The specific countermeasures he proposes are neighbor-to-neighbor digital signature of routing updates, the addition of sequence numbers and timestamps to the updates, and the addition of acknowledgments and retransmissions of routing updates. These results are similar to those presented here with the exception that the countermeasures presented here explicitly assume the existence of subverted routers, and provide countermeasures to protect against them. This is important as routers are potentially vulnerable to attack from a number of sources, with potentially catastrophic results from success.

Murphy [12] outlines a solution for securing distance-vector protocols that involves including the information used to select a route, signed by the neighbor it received it from, in the routing update it then signs and transmits to its neighbors. She points out that this requires the validation of a number nested signatures equal to the number of routers in the path. This results in both update size and validation computation time problems as the size of the network grows. These problems result, fundamentally, from the redundant signing of link information for paths that are supersets of paths used to reach destinations traversed in the longer path. The countermeasures presented here avoid these problems by signing only the component link information, in the form of predecessors, and performing a path traversal to validate full paths. This results in the use of constant space, and significantly reduced computation time.

Smith and Garcia-Luna-Aceves [20] have analyzed the Border Gateway Routing Protocol (BGP) in a manner similar to the analysis presented here. BGP is a member of the path-vector class of routing protocols, which carry full path information in the routing updates to allow loop detection, and the use of non-uniform route selection metrics. The solutions developed for BGP are similar to the ones presented here in that they use the cryptographic protection of the first hop information in the path by the destination (originating) router.

Chapter 7

Concluding Remarks

This document presents an analysis of the security of distance-vector routing protocols. It identifies vulnerabilities in this class of protocols that could potentially result in the deception or disruption of the routing computation, or the disclosure of sensitive routing information. It then presents a set of countermeasures which is the first solution known to protect these protocols at a cost and level of effectiveness equivalent to that of current link-state security proposals. Specifically, this solution protects the distance-vector routing protocols from all classes of intruders — masquerading routers, unauthorized routers, subverted links, and subverted routers — at a constant cost per destination in an internet. This solution introduces the innovation of using authenticated predecessor information to reconstruct the path implied by an update. In summary, the solution presented here shows that distance-vector protocols can be secured at levels of efficiency and effectiveness equivalent to current proposals for link-state protocol security.

A number of data structures are defined for use in the pseudo-code for the distance-vector algorithms defined below.

Sequence Number ($seqNum$): The sequence number maintained by each router.

Sequence Number Table (SN^i_{jm}): The Sequence Number table maintained at node i contains the largest sequence value seen in a routing update with originating router m for destination network j.

Link-Cost Table (L_i): The Link-Cost table maintained at node i describes the networks node i is attached to. Each entry includes the following information:

l_n → the cost of the link to the attached network n. The cost of a failed link or a link to a failed network is infinity.

$lmod_n$ → a boolean indicating whether this entry has been modified

Update Message (U_k): Each update, U_k, from received by router i from neighboring router k is a column vector of update entries with the following fields:

j → destination

UD^k_j → distance from k to j

up^k_j → predecessor network

usn^k_j → update sequence number

ul^k_j → link cost of j

uds^k_j → digital signature information protecting the destination, predecessor, destination link cost, and sequence number information as computed by the originator — this will be a complex data structure including information appropriate to the digital signature solution implemented (see Section 5.2.4)

Distance Table (DT_i): The Distance Table at router i is a matrix containing, for each destination network j and neighboring router k, the following information regarding the route reported by k:

D^i_{jk} → distance from k to j

p^i_{jk} → predecessor network

sn^i_{jk} → update sequence number

l^i_{jk} → link cost of j

ds^i_{jk} → digital signature information protecting the destination, predecessor, destination link cost, and sequence number information as computed by the originator — this will be a complex data structure including information appropriate to the digital signature solution implemented (see Section 5.2.4)

Routing Table (RT_i): The Routing Table at router i is a column vector of entries for each known destination network j which specify the following regarding the routes chosen by i:

D^i_j → distance from i to j

p^i_j → predecessor network

s^i_j → successor router

sn^i_j → update sequence number

l^i_j → link cost of j

ds^i_j → digital signature information protecting the destination, predecessor, destination link cost, and sequence number information as computed by the originator — this will be a complex data structure including information appropriate to the digital signature solution implemented (see Section 5.2.4)

$RTmod^i_j$ → a boolean indicating whether this entry has been modified

In addition a number of routines are called in the pseudo-code, but not defined.

DigSig(j, p, sn, l, x): This routine returns the digital signature information for the destination network j, predecessor router p, sequence information sn, and destination link cost l for originating router x. The specific digital signature algorithms used and information returned depends on the specifics of the particular implementation, as described in the text.

Network(x): This routine returns the attached network from L_i that is shared with the neighboring router with address x.

Originator(ds): Extracts and returns the id of the originating router from the digital signature information.

SelectRoute(i, j): This routine picks the preferred route from router i to destination network j among the available routes with the highest sequence number. The specifics of how this decision is made depends on the particular implementation.

TransmitUpdate(k, U): This routine transmits the update U to neighbor k.

Figure 7.1: Definition of Primitives for Secure Distance-Vector Processing

```
procedure LinkChange($i, n, c$)
when router $i$ detects a change of its link to network $n$ to cost $c$
begin
    $l_n \leftarrow c$;
    $lmod_n \leftarrow$ true;
    call UpdateRT($i$);
end

procedure ReceiveUpdate($U_k$)
when router $i$ receives an update $U_k$ from router $k$
begin
    for each update entry $(j, UD_j^k, up_j^k, usn_j^k, ul_j^k, uds_j^k)$ in $U_k$ do
    begin
        $o \leftarrow$ Originator($ds$);
        if $((usn_j^k \geq SN_{jo}^i)$ and $(ds =$ DigSig$(j, p, sn, l, o)))$
        then begin
            $D_{jk}^i \leftarrow UD_j^k; p_{jk}^i \leftarrow up_j^k; sn_{jk}^i \leftarrow usn_j^k; l_{jk}^i \leftarrow ul_j^k;$
            $ds_{jk}^i \leftarrow uds_j^k; SN_{jo}^i \leftarrow usn_j^k;$
        end
    end
    call UpdateRT($i$);
end

function ValidatePath($i, k, d, p, l$) → boolean;
begin
    $tj \leftarrow p; p \leftarrow p_{tj,k}^i; td \leftarrow l;$
    while $((p\ not\ in\ L_i)$ and $(p \neq null))$ do
    begin
        $td \leftarrow td + l_{tj,k}^i; tj \leftarrow p; p \leftarrow p_{tj,k}^i;$
    end
    if $(p\ in\ L_i)$
        then return $((td + l_p) = d)$;
        else return false;
end

procedure UpdateRT($i$)
begin
    for each destination $j$ in $DT_i$ do
    begin
        repeat
            $(D_{jx}^i, p_{jx}^i, sn_{jx}^i, l_{jx}^i, ds_{jx}^i) \leftarrow$ SelectRoute$(i, j)$;
        until $((x =$ null$)$ or ValidatePath$(i, x, D_{jx}^i, p_{jx}^i, l_{jx}^i))$;
        if $((x \neq$ null$)$ and $((D_j^i \neq D_{jx}^i)$ or $(s_j^i \neq x)$ or $(ds_j^i \neq ds_{jx}^i)))$
        then begin
            $D_j^i \leftarrow D_{jx}^i + l_{Network(x)}; s_j^i \leftarrow x; p_j^i \leftarrow p_{jx}^i;$
            $sn_j^i \leftarrow sn_{jx}^i; ds_j^i \leftarrow ds_{jx}^i;$
            $RTmod_j^i \leftarrow$ true;
        end
        else if $(x =$ null$)$ then error "No valid route to destination $j$.";
    end
    call SendUpdates($i$);
end

procedure SendUpdates($i$)
begin
    for each destination $j$ in $RT_i$ where $RTmod_j^i =$ true do
    begin
        $U_{tmp} \leftarrow U_{tmp} \cup (j, D_j^i, p_j^i, sn_j^i, l_j^i, ds_j^i)$;
        $RTmod_j^i \leftarrow$ false;
    end
    for each attached network $j$ in $L_i$ do
        call TransmitUpdate($U_{tmp}$);
    $sn \leftarrow seqNum; seqNum \leftarrow seqNum + 1;$
    for each attached network $j$ in $L_i$ where $lmod_j =$ true do
    begin
        for each attached network $p$ in $L_i$ where $p \neq j$ do
            call TransmitUpdate($p$, $(j, l_j, p, sn, l_j,$ DigSig$(j, p, sn, l, i)))$;
        $lmod_j \leftarrow$ false;
    end
end
```

Figure 7.2: Pseudo-code for Secure Distance-Vector Processing

Bibliography

[1] Randall Atkinson. Security Architecture for the Internet Protocol. RFC 1825, August 1995.

[2] Dimitri Bertsekas and Robert Gallager. *Data Networks.* Prentice Hall, 2nd edition, 1992.

[3] Donald E. Eastlake 3rd and Charles W. Kaufman. Domain Name System Security Extensions. Internet draft: draft-ietf-dnssec-secext-10.txt, August 1996.

[4] Gregory G. Finn. Reducing the Vulnerability of Dynamic Computer Networks. Technical Report ISI/RR-88-201, Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, CA 90292-6695, June 1988.

[5] J. J. Garcia-Luna-Aceves and Shree Murthy. A Path-Finding Algorithm for Loop-Free Routing. *IEEE/ACM Transactions on Networking*, 5(1):148–160, February 1997.

[6] Ralf Hauser, Tony Przygienda, and Gene Tsudik. Reducing the Cost of Security in Link-State Routing. In *Proceedings: Symposium on Network and Distributed System Security*, pages 93–99. Internet Society, February 1997.

[7] Laurent Joncheray. A Simple Active Attack Against TCP. In *Proc. 5th UNIX Security Symposium*, pages 7–19. The USENIX Association, June 1995.

[8] Donald E. Knuth. *The Art of Computer Programming: Volume 1, Fundamental Algorithms.* Addison-Wesley, 2nd edition, 1973.

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

[10] G. Malkin. RIP Version 2: Carrying Additoinal Information. RFC 1723, November 1994.

[11] Robert T. Morris. A Weakness in the 4.2BSD Unix TCP/IP Software. Technical Report 117, AT&T Bell Laboratories, Murray Hill, New Jersey 07974, February 1985. ftp://netlib.att.com/netlib/att/cs/cstr/117.ps.Z.

[12] Sandra L. Murphy. Digital Signature Protection of the OSPF Routing Protocol. *Proceedings: Symposium on Network and Distributed System Security*, 1996. http://bilbo.usy.edu/sndss/sndss96.html.

[13] Shree Murthy and J. J. Garcia-Luna-Aceves. A more efficient path-finding algorithm. In *28th Asilomar Conference*, pages 229–233, Pacific Grove, California, November 1994.

[14] Dan Nessett. The Internet Security Architecture. In *Proceedings: Internet Security Workshop.* IEEE, November 1994.

[15] R. Perlman. *Network Layer Protocols with Byzantine Robustness.* Report MIT/LCS/TR 429, Massachusetts Institute of Technology, October 1988.

[16] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4). RFC 1771, March 1995.

[17] Eric C. Rosen. Vulnerabilities of Network Control Protocols: An Example. RFC 789, 1994.

[18] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc, 2nd edition, 1996.

[19] Robert W. Shirey. Security Architecture for Internet Protocols. A Guide for Protocol Designs and Standards. Internet Draft: draft-irtf-psrg-secarch-sect1-00.txt, November 1994.

[20] Bradley R. Smith and J. J. Garcia-Luna-Aceves. Securing the Border Gateway Routing Protocol. *Proceedings of Global Internet '96*, November 1996.

[21] Bradley R. Smith, Shree Murthy, and J. J. Garcia-Luna-Aceves. Securing Distance-Vector Routing Protocols. In *Proceedings: Symposium on Network and Distributed System Security*, pages 85–92. Internet Society, February 1997.