UNIVERSITÄT
KOBLENZ · LANDAU

Fachbereich 4: Informatik

Security in Distance Vector Algorithms

by

Stefanie Lück

A thesis submitted to the
Faculty of Computer Science
at the University of Koblenz–Landau
in partial fulfillment of the requirements
for the degree of
Diplom-Informatiker

Reviewer:	Prof. Dr. Christoph Steigner,
	University of Koblenz–Landau, Faculty of Computer Science
Supervisor:	Dipl.-Inform. Frank Bohdanowicz,
	University of Koblenz–Landau, Faculty of Computer Science

September 2011

Ehrenwörtliche Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Arbeit wurde noch keiner Prüfungsbehörde in gleicher oder ähnlicher Form vorgelegt.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

. . . .

(Ort und Datum) (Unterschrift)

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

English Distance vector routing protocols are interior gateway protocols in which every router sets up a routing table with the help of the information it receives from its neighboring routers. The routing table contains the next hops and associated distances on the shortest paths to every other router in the network.

Security mechanisms implemented in distance vector routing protocols are insufficient. It is rather assumed that the environment is trustworthy. However, routers can be malicious for several reasons and manipulate routing by injecting false routing updates.

Authenticity and integrity of transmitted routing updates have to be guaranteed and at the same time performance and benefits should be well-balanced.

In this paper several approaches that aim at meeting the above mentioned conditions are examined and their advantages and disadvantages are compared.

German Distanzvektor-Routing-Protokolle sind Interior-Gateway-Protokolle, bei denen jeder Router anhand der Informationen, die er von seinen Nachbarn erhält, eine Routingtabelle mit den kürzesten Wegen und dazugehörigen Kosten zu allen anderen Routern des Netzwerks aufbaut.

Distanzvektor-Routing-Protokolle sehen jedoch nur unzureichende Mechanismen vor, um die Sicherheit ihrer Operationen zu gewährleisten. Es wird vielmehr einfach davon ausgegangen, dass die Umgebung vertrauenswürdig ist. Router können sich aber aus verschiedenen Gründen böswillig verhalten und falsche Routingupdates einschleusen um das Routing zu manipulieren.

Authentizität und Integrität der übermittelten Routinginformationen müssen daher sichergestellt werden; dabei soll eine Balance zwischen Nutzen und Performance gefunden werden. Diese Arbeit untersucht verschiedene Lösungsansätze, die sich die Erfüllung dieser Anforderungen zum Ziel gesetzt haben, und stellt deren Vor- und Nachteile einander gegenüber.

1.2 Thesis Structure

This thesis is organized as follows: Chapter 2 gives an overview on the fundamental principles of routing and the weak points that especially distance vector routing has to face. Digital signatures and MD5, a message digest algorithm, are outlined as their understanding is required in the course of the thesis.

Chapter 3 presents the approaches to securing distance vector routing protocols that have been considered useful during literature research. It starts with RIPv2's own authentication mechanism, followed by an approach that uses digital signatures and sequence numbers as well as a path finding algorithm to recursively reconstruct a route. RIP (Routing Information Protocol) with triangle theorem checking and probing, the third solution, uses a so-called *triangle theorem* and probing messages to detect malicious updates. Algorithm four, a pivot based algorithm for inconsistency recovery (PAIR), adds new metrics to a routing update that are used for discovering inconsistencies. With the help of certain mathematical properties it can even recover from false updates. Special features of algorithm five, S-RIP (secure RIP), are a reputation-based framework that assigns nodes a trust level and treats them accordingly as well as the use of consistency checks that are used to validate routes with the routers that have propagated the route. Approach six, S-DV (secure distance vector routing protocol), also uses consistency checks but tries to make them more effective by introducing trusted routers. In the last solution that is examined, a symmetric key based approach, each router in a network owns a set of symmetric keys of which it shares a unique subset with every other router in the network. The keys are used to sign routing updates.

Chapter 4 briefly sums up the main characteristics of each approach and looks into their complexity and overhead. As far as simulations have been performed by the authors the results are presented here. Furthermore, the advantages and disadvantages of each technique are compared. A table at the end of the chapter summarizes the results.

Finally, future work is discussed in Chapter 5.

2 Theoretical Principles

This chapter conveys basic information that is needed in the course of this paper. In the first section, the concepts of routing in general and distance vector routing protocols in particular are explained. The next topic to be examined are problems that might occur in distance vector routing. Finally, digital signatures and MD5, a cryptographic hash function that can be used for authentication in RIP, are presented.

2.1 Routing Protocols

Routing is one of the important tasks the Network Layer of the OSI model has to accomplish. In this layer data packets are forwarded from a source to a destination. For that purpose a router has to know which path to send a data packet on. This decision is made with the help of a *routing table*; the process of constructing the routing table with the help of routing algorithms is called *routing.* There is a weight assigned to each link between two nodes. In simple terms routing protocols help finding the least expensive route between two nodes (hosts, routers or networks). Here, "least expensive" means the lowest sum of edge weights. Unless otherwise indicated edges are assigned a weight of 1 in the following.

When a packet arrives in a router, the router does not yet know how to deliver the packet to its destination. It uses the help of routing tables to choose the optimal next hop for this particular packet's destination address. This second step is called *forwarding.* It is important to distinguish between routing and forwarding, although the difference between these terms is often neglected [PD07].

There are two types of routing: *Interdomain routing* (using *exterior gateway protocols*), which is applied in routing *between* autonomous systems, and *intrado-*

main routing (using *interior gateway protocols*), which is applied in routing *in* autonomous systems. Intradomain routing can be further classified into *link state routing* and *distance vector routing*, this paper focuses on the latter. While in link state routing (for example OSPF, see [Moy98]) routers use the help of *reliable flooding* [PD07] to propagate information about their directly connected neighbors and associated costs to the whole network, in distance vector routing a router imparts only to its neighbors what the network looks like from its own point of view.

2.1.1 Distance Vector Routing Protocols

Distance vector routing (also named *Bellman-Ford-Routing*, after its developers [Tan03]) is an interior gateway protocol for exchanging routing information in autonomous systems. The term is derived from the fact that in distance vector routing a router works with vectors (one-dimensional arrays) that store the distances to all the other nodes in the network. One of the most used distance vector routing protocols is RIP (Routing Information Protocol). Its core algorithm was first used in the ARPANET already in 1969 [Hed88]. After RIPv1 and RIPv2 the current version is RIPng (RIP next generation). In the following, distance vector protocols will be examined using the example of RIP.

RIP example: Figure 2.1 shows a network consisting of six routers and their links. It is assumed that the weight of each link corresponds to one. That means hop count is used as a metric, the "cheapest" path is always the one with the lowest number of hops. At the beginning, every router only knows that the distance to its directly connected neighboring routers is one (and as a consequence, that the next hop is the neighbor itself) and that it cannot reach any other router yet, which is denoted by setting these distances to ∞. Table 2.1 shows router C's knowledge before the algorithm starts.

Router A, B, D, E and F set up their initial routing table accordingly. Note that none of the routers has global knowledge, each of them only knows its own routing table. For example, router A is not aware of D's distance to F. The aim of the routing algorithm is to reach a stable state as soon as possible. This can be either after topology changes or after start-up of the algorithm. As soon

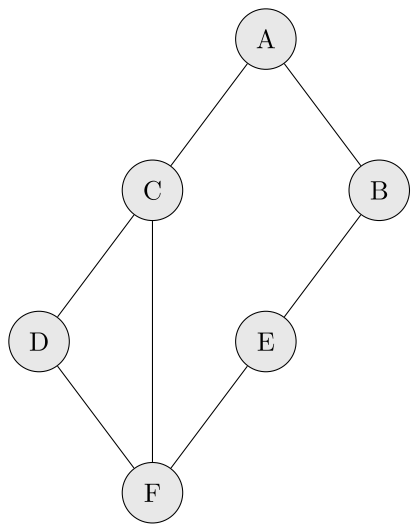


Figure 2.1: An example network consisting of 6 routers

destination	cost	next hop
A	1	A
B	∞	-
D	1	D
E	∞	-
F	1	F

Table 2.1: Initial routing table for router C

as a stable state is reached, this is referred to as *convergence*. The smaller the network, the faster convergence will be achieved.

Every router sends its own distance vector, i.e. information about the current costs and next hops to any other router in the network, to its neighbors (and *only* to its neighbors!).

In our example, router A tells router C that it can reach router B in one hop. So far, C only knew a distance of ∞ to B. As C knows that A is its neighbor (and thus it can reach A in one hop), it calculates that it can reach B in two hops. This is less than the previous distance which was set to ∞. Hence, C updates its routing table entry for B to distance two and next hop A. C also learns from F that F is one hop away from D and concludes that it can reach D in two hops via F. But C already knows D as its direct neighbor, which means it can be reached in only one hop. Therefore, it ignores this information and

	A		**B**		**C**		**D**		**E**		**F**	
	cost	nh	cost	nh	cost	nh	cost	nh	cost	nh	cost	nh
A	0	-	1	B	1	C	2	C	2	B	2	C
B	1	A	0	-	2	A	3	A[1]	1	E	2	E
C	1	A	2	A	0	-	1	D	2	F	1	F
D	2	C	3	C[2]	1	C	0	-	2	F	1	F
E	2	B	1	B	2	F	2	F	0	-	1	F
F	2	C	2	E	1	C	1	D	1	E	0	-

Table 2.2: Global routing table after convergence in RIP

leaves the corresponding routing table entry unaltered. The same applies to D if it sends its neighbor C a distance of 1 to reach F.

Table 2.2 shows each router's routing table entries after routing is convergent. Note again that none of the routers has the global view of the network that is depicted in the table; each router only knows the entries in its row.

Routing updates: There are two occasions in which routing updates are sent. First, updates are sent periodically (every 30 seconds in RIP), even if the topology has not changed; by doing so a router informs its routing domain that it is still "alive". Second, updates are sent after a router had to change one of its routing table entries. It then sends out updates to its neighbors who in turn can change their routing tables if necessary and so on. This is called *triggered update* and proceeds until convergence is again accomplished.

Recovery in RIP: If routing updates from another router are missing for several cycles, the affected router concludes that the link connecting them is damaged or that the other router is not working anymore. Another way to discover this is checking links by sending out control packets that have to be confirmed by the other router. Once a broken link or node is discovered the revealing node sends out new routing updates so that a stable state can eventually be reached. The following example illustrates this situation: Assuming A finds out that its link to router C is broken, it changes its routing table and sets the

[1]Next hop can also be E, depending on which route was announced first.
[2]Next hop can also be F, depending on which route was announced first.

distance to C to ∞ instead of 1 and propagates the new distance to B. B used to have a route with next hop A and distance 2 to reach C and concludes that it has to change this distance to ∞. Soon, B gets an update from E that tells it E has a distance of 2 to reach C. B knows that it is directly connected to E and deduces that it can reach C in 3 hops. This is less than ∞, so the entry is updated. A is informed by B that B can now reach C in 3 hops and as B is its neighbor, A updates its route to C and can now reach C in 4 hops via B. Although the link between C and A is broken A has now learned a new route to reach C and the system has converged.

Count-to-infinity: In case routing updates arrive in an unfavorable order a problem called *count-to-infinity* can occur. This can happen when a link between two nodes breaks down. The crux is that "good news travels quickly, bad news travels slowly" [Lin04], meaning that distance vector algorithms react quickly to new shorter routes that are propagated whereas they need a long time to stabilize after a link breaks or a router fails. In fact, in a network whose longest path consists of n links it takes n update cycles to inform every router about newly added routers or links. In contrast to this it takes much longer to inform all the nodes in a network about disconnection of routers or links. In a particular router the distance to a certain destination is at least up by one hop count compared to the minimum of its neighbor's distances to that destination. It takes some time for all routers to count to infinity and to conclude that a destination is unreachable.

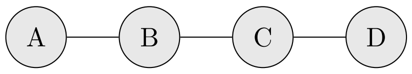


Figure 2.2: Example for count-to-infinity

If the link between router A and router B (see Figure 2.2) becomes inoperative, B announces a distance of ∞ to A while C announces a distance of 2 to A (since it is not aware of the broken link between A and B). B receives C's update and thinks that there is another route from C to A which it might use. Thus, B calculates that it can reach A in 3 hops (1 hop from B to C, as they are directly connected, plus 2 hops for the new route it has just learned). When C receives the next update from B it learns that B has changed its cost to reach A from

1 to 3. C concludes that it has to update its distance to reach A from 2 to 4 (1 hop to reach B and 3 hops to reach A from B, as it has just learned). This escalation continues until a distance is reached that is considered to be infinite. The system cannot converge as none of the routers knows that A is unreachable.

There are several approaches to solving the count-to-infinity problem. For example in RIP, infinity is set to 16 (maximum hop count + 1), which represents unreachability of a destination. As a result, count-to-infinity does not take that long, but problems can arise if there are routes in the network that consist of more than 15 hops. Another approach to a solution is called *split-horizon*: When a router forwards routing updates it never forwards a route to the router itself has learned it from. In the above example router C is not allowed to advertise its route for A back to B. This might seem redundant as a route from B via C to A would be a detour. But in case the link that connects A and B broke down and C had advertised a route to B, B could decide to use that route. A routing loop would be created as C would send the information back to B and so on. To enforce the effect of split-horizon, *split-horizon with poison reverse* can be used. In this modification a router *does* announce a route back to where it was learned from, but sets the distance to ∞. By making this route look unappealing it assures that the other router will not use it. However, there is a drawback in split-horizon as well as in split-horizon with poison reverse: they both only remedy deficiencies if the loop is made up of exactly two nodes. Figure 2.3 shows a Y-topology that includes a routing loop made up of three routers, A, B and C. The following explanation illustrates why split-horizon cannot prevent from count-to-infinity if the sequence of routing updates is disadvantageous. Before the link between node C and node D breaks, node A and node B each hold a route in their routing tables with destination D, next hop C and distance 2. C holds a route to D with next hop D and distance 1. Now assume that the link between node C and node D breaks and D is cut off from the network. In an ideal case, A and B would receive C's new routing update, that states the unreachability of D, at the same time. But if A receives it before B gets it, A updates its routing table entry for destination D as unreachable, whereas B is not yet aware of D's unreachability and next sends A an update for destination D with its “old” distance 2 and next hop C. Meanwhile, B receives C's message, stating that D is unreachable. At this point, A thinks that it can reach D in 3

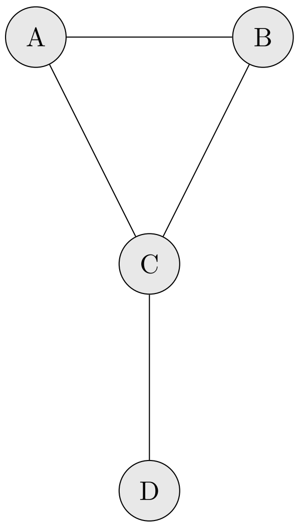


Figure 2.3: Split-horizon does not prevent count-to-infinity in a Y-topology

hops via *B*. It sends this route to *C*, but not to *B*, as split-horizon forbids that it sends it back to where it has learned it from. Thus, *C* thinks that it can reach *D* in 4 hops via *A* and sends this route to *B*, but not back to *A*, as it has learned it from *A*. *B* concludes that it can reach *D* in 5 hops via *C* and sends this route to *A*, but not back to *C*. *A* concludes that it can reach *D* in 6 hops via *B* and sends this route to *C*, but not back to *B*. This cycle continues until every router's distance to *D* reaches ∞. Table 2.3 shows the chronology of routing updates for routers *A*, *B* and *C*. As can be seen, count-to-infinity can still occur if there are loops that consist of 3 or more nodes, even if split-horizon is implemented.

	A			**B**			**C**		
	dest	dist	nh	dest	dist	nh	dest	dist	nh
Step 1	B	1	B	A	1	A	A	1	A
	C	1	C	C	1	C	B	1	B
	D	2	C	D	2	C	D	∞	–
Step 2	B	1	B	A	1	A	A	1	A
	C	1	C	C	1	C	B	1	B
	D	∞	–	D	2	C	D	∞	–
Step 3	B	1	B	A	1	A	A	1	A
	C	1	C	C	1	C	B	1	B
	D	3	B	D	2	C	D	∞	–
Step 4	B	1	B	A	1	A	A	1	A
	C	1	C	C	1	C	B	1	B
	D	3	B	D	∞	–	D	4	A
Step 5	B	1	B	A	1	A	A	1	A
	C	1	C	C	1	C	B	1	B
	D	3	B	D	5	C	D	4	A
Step 6	B	1	B	A	1	A	A	1	A
	C	1	C	C	1	C	B	1	B
	D	6	B	D	8	C	D	7	A
Step 7	B	1	B	A	1	A	A	1	A
	C	1	C	C	1	C	B	1	B
	D	9	B	D	11	C	D	10	A
Step 8	B	1	B	A	1	A	A	1	A
	C	1	C	C	1	C	B	1	B
	D	12	B	D	14	C	D	13	A
Step 9	B	1	B	A	1	A	A	1	A
	C	1	C	C	1	C	B	1	B
	D	15	B	D	∞	–	D	∞	–
Step 10	B	1	B	A	1	A	A	1	A
	C	1	C	C	1	C	B	1	B
	D	∞	–	D	∞	–	D	∞	–

Table 2.3: Sequence of routing updates for count-to-infinity in a Y-topology

2.1.2 RIP - Packet Format

This section describes the structure of RIP packets in RIPv1 and RIPv2 in order to make the use of implemented authentication mechanisms in RIPv2 easier to

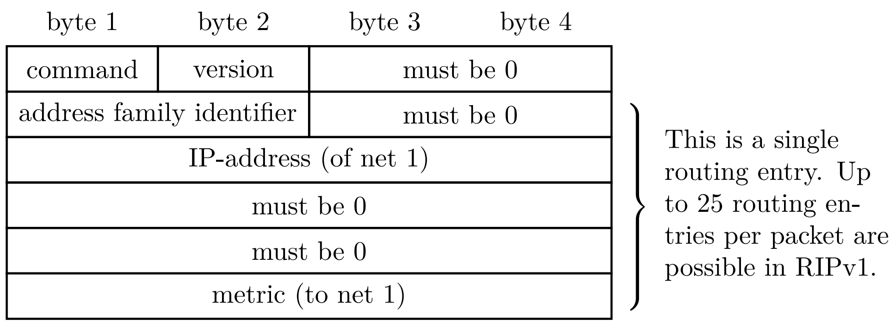

byte 1
byte 2
byte 3
byte 4
command
version
must be 0
address family identifier
must be 0
IP-address (of net 1)
must be 0
must be 0
metric (to net 1)
This is a single routing entry. Up to 25 routing entries per packet are possible in RIPv1.

Figure 2.4: Format of a packet in RIPv1

understand. Section 3.1 puts a finer point to authentication in RIPv2. Figure 2.4 shows the format of a packet in RIPv1. In the following, the fields of a RIPv1 packet are explained:

- *command*: Defines the intention of the packet. The main purposes are
  - *request*: Asks the receiver to send its routing table or parts of it.
  - *response*: Is an update message or a response to a request. It contains the sender's routing table or parts of it.
- *version*: Specifies the current RIP-version.
- *address family identifier*: RIP can carry information for different protocols. The address family identifier indicates the type of address that is specified in a particular entry. For IP, the address family identifier is 2.
- *IP-address*: In RIP, routes can be described to certain hosts or to networks. The 32 bit address in this field is either a host address, a subnet number or a network number. If the entry is 0, this indicates a default route. A packet's destination address is first checked against the list of host addresses, then against the list of subnet numbers and finally against the list of network numbers. This procedure assures that the most specific information is used.
- *metric*: This field contains a value between 1 and 15 and specifies the metric for a destination. A metric of 16 means that the destination is unreachable.

Because of some downsides in RIPv1 a new version of RIP, RIPv2, was developed in 1993 [Mal93]. It is practically an extension of RIPv1 that adds two important features: It supports classless interdomain routing (CIDR) by including the subnet mask in a routing entry, furthermore cleartext authentication is now possible. In 1997 authentication was improved by introducing MD5 authentication, which is further examined in Section 3.1. These new functions call for a change in packet format, as more information needs to be stored. The header of a RIP package remains unaltered whereas the 20 byte route entry is modified. Figure 2.5 shows the new packet format of RIPv2 [Mal98].

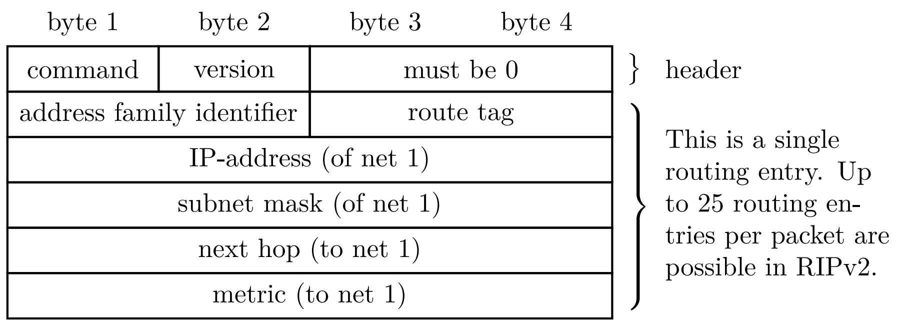


Figure 2.5: Format of a packet in RIPv2

- *version*: If one of the newly defined fields is filled out or if the message uses authentication, the version field contains the number 2.
- *route tag*: Distinguishes between internal routes (learned by RIP) or external routes (learned from other protocols).
- *subnet mask*: Contains the destination's subnet mask and helps to determine the non-host part of the address.
- *next hop*: Specifies the next hop IP-address on the packet's way to the destination. This entry can be ignored but in this case an optimal route is no longer guaranteed.

If authentication is not used, a RIPv2 packet looks like depicted in Figure 2.5 and can carry up to 25 routing entries. If, however, authentication is used, the packet's structure must be modified. Since there are only two "unused" bytes in a RIPv2 message header which could be used for authentication and since this is not sufficient, the space of a complete routing entry (20 bytes) is used for authentication [Mal98]. This reduces the amount of possible routing entries to 24. A RIPv2 message that uses authentication can be identified by the entry *0xFFFF* in the address identifier field. The next two bytes contain the authentication type (2 for cleartext password), this is followed by the authentication key (the plaintext password), which consists of 16 bytes. Figure 2.6 shows a RIPv2 packet that makes use of authentication.

byte 1	byte 2	byte 3	byte 4
command	version	must be 0	
0xFFFF		authentication type	
authentication key			
authentication key			
authentication key			
authentication key			
up to 24 routing entries at 20 bytes			

Figure 2.6: Format of a packet that uses authentication in RIPv2

As RIP is UDP-based, routing messages are sent and received on UDP port number 520. There are two exceptions: It is possible to send specific queries from a different port, but in that case they must be sent to a UDP port. If a message is the response to a request, it is sent to the port from which the request was sent.

2.2 Vulnerabilities in Distance Vector Routing Protocols

Provided techniques for authentication in distance vector algorithms are weak and it is almost unfeasible to validate routing updates. RIP, for example, only checks if an incoming routing update was sent by a neighboring router and – in case it makes use of authentication techniques – if the received update's data origin and data integrity are accurate. While alteration of messages can be detected, factual incorrectness of a routing update remains undetected. Therefore, distance vector protocols are exposed to various threats and attacks. The most important ones [WKV04] are discussed in this section. A fatal problem in distance vector routing protocols is that routers implicitly trust each other. Even if only a single router in a network is malicious or malfunctioning, routing protocols can be severely violated [Per92]. This weakness leads to the need for solutions to the following problems that can occur. Several promising approaches are discussed in Chapter 3.

Shorter distance fraud: A malicious router claims to have a shorter distance to another router or subnet than it actually has. By this means it becomes more attractive for neighboring routers to use this particular router as their next hop on the way to a certain destination, although the distance is in fact not shorter at all. An example for shorter distance fraud is given in Figure 2.7.

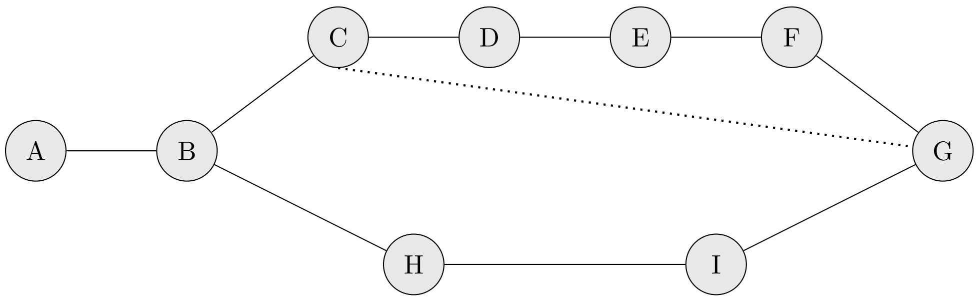


Figure 2.7: Example for shorter distance fraud

Router *C* needs four hops to reach router *G*, but claims to be able to reach *G* in only one hop (depicted by the dotted line). Thus, *B* will send packets destined for *G* via *C* instead of *H*. Taking the route via *H* would in fact be shorter (three hops), but as *B* believes *C*, traffic is redirected. There can be several reasons for this kind of attack. The malicious router attracts traffic and then it can just drop the messages it gets so they will never reach their destination. This is called a *black hole*. It can also eavesdrop on the messages to gain knowledge it is not entitled to get. Moreover, the router claiming a shorter distance can alter messages or even inject packages [HPJ03]. It is obvious that the whole topology can be easily disrupted if a router uses shorter distance fraud.

Longer distance fraud: Another popular way to compromise routing is longer distance fraud. This means that a deceiving router tries to keep traffic off itself by pretending to be farther away from certain destinations than it really is. Other routers will not use the malicious router as a next hop to certain destinations as the route through it seems to be longer than other routes. In Figure 2.7 router H could claim to be five hops away from G. B would then send its packets destined for G via C, as this route seems to be shorter. Actually, if B chooses C as next hop on its route to G, packets have to travel five hops, which is two hops more than the route via H. Avoidance of traffic can be a means of reducing costs and it can also lead to an unjust use of network connections. In the worst case the network can even get congested [WKV04].

Router impersonation: As the name already suggests, router impersonation means that a router that is not authorized takes part in routing operations. In RIPv1 this is particularly easy as there are no authentication tools at all. RIPv2 provides for cleartext authentication, which only helps ostensibly as it can be easily subverted. Authentication in RIPv2 can also be secured by using keyed MD5 (see Sections 2.4 and 3.1). However, MD5 is no longer considered safe nowadays.

Prefix impersonation: If a dubious router claims to have a distance of zero to either a non-existent network or a network that is not directly connected, this is called prefix impersonation [WKV04]. The use of MD5 authentication is of no avail here, so prefix impersonation can severely disrupt distance vector routing protocols.

Chakrabarti and Manimaran [CM03] differentiate between router attacks and link attacks. In router attacks, a malevolent router manipulates distance vector data. This router can either be the source of the update or it can be located somewhere between source and destination of the update. Link attacks are different insofar as an update is altered illegally by an imposter who successfully gains access to a link in the network.

Smith [Smi97] defines threats as potential violations of security and divides them into four classes:

- Disclosure: A non-authorized device gains access to data.

- Deception: An authorized entity gets false information and believes it to be true.
- Disruption: The correct functioning of a system is disturbed or precluded.
- Usurpation: An unauthorized entity finally controls system services.

Smith notes that threats and vulnerabilities (weaknesses that can be exploited) can be reduced by providing six security services:

- Confidentiality: Protection of data from unauthorized disclosure.
- Integrity: Data is not altered on its way from sender to recipient.
- Authenticity: It can be verified that an entity's identity is correct.
- Access control: Only entities that are entitled to access a system are granted access.
- Non-repudiation: A sender cannot deny having sent certain data and a receiver cannot deny having received certain data.
- Availability: A system should be available for usage at any time.

By using countermeasures like encryption or digital signatures security services can be provided. They help against four types of intruders that can construct, observe, delete, alter or replay routing information. The first type of intruders are *masquerading routers*, which are routers that imitate identities of authorized routers. The second type are *subverted routers*, which are routers that are brought to brake routing protocols or to illegally lay claim to network resources. This can happen if there are errors in the configuration information or in the routing code or if a router was tricked into loading incorrect configuration information or software. Thirdly, there are *unauthorized routers*. An unauthorized router succeeds in taking part in the routing process by evading access control methods. Lastly, *subverted links* are links that are controlled from the outside either through access to the physical medium or by interfering with the protocols that are underlying the routing protocol (e.g. the TCP session hijacking attack) [Smi97].

Smith's classification into four types of intruders and four types of general threats leads to specific endangerments to routing data. Attacks take advantage of missing authentication mechanisms, missing access control mechanisms and the fact that integrity of routing updates cannot be assured. A deceiver that successfully enters a routing domain which it is not entitled to enter has several means of manipulating or destroying the routing process [Smi97]. By replaying or deleting routing messages an intruder can set back a network's configuration to a previous state. The intruder can also cause revelation of network traffic, denial of service in the whole network or faulty computation of network resource usage. It merely has to rearrange the topology by altering or inventing routing messages.

2.3 Digital Signatures

This section gives a short introduction to digital signatures, which are used by some approaches to secure distance vector algorithms. Digital signatures help to assure integrity, authenticity and non-repudiation of messages, whereas they do not guarantee confidentiality. This means the receiver of a digitally signed message can make sure that the message was

- actually sent by the claimed originator,
- that the sender cannot deny having sent the message
- and that the message's content was not altered on its way from sender to receiver.

If the sender does not make use of encryption in addition to signing the message, the message is not confidential and can be read by everyone.

Digital signatures are based on asymmetric cryptography, meaning that they use a pair of keys, namely *private key* (secret) and *public key* (not secret). Every participant has his own pair of keys. The private key is used to calculate a digital signature, the public key is used to verify the signature. More precisely, the signature is computed from the sender's private key and the original message using a unique calculation specification or signing algorithm. The most popular signing algorithm is RSA (see [JK03] for details); the fact that it is very difficult to decompose very large numbers into their prime factors makes it secure. Other examples are ElGamal (see [EG85]) and DSA (Digital Signature Algorithm, see [fip09]). In order to ensure the correct functioning of digital signatures the signing algorithm must produce different signatures as an output for different messages it gets as an input with a probability bordering on certainty.

There is one problem that arises if signatures are computed based on the complete original message: As the signature is an encoded version of the message itself, the signed message will be roughly twice as big as the original message [Smi97]. This is obviously inefficient and results in the use of hash functions. Hash functions map a message onto a *message digest* or *fingerprint* that has a fixed length and is usually significantly smaller than the message itself but still identifies the message with very high probability (for an example of a hash algorithm see Section 2.4). A good hash function is supposed to be collision-free, which means that different inputs produce different outputs. If now the signature is computed from the message digest instead of the much bigger original message, the procedure is more effective and efficient.

Figure 2.8 shows the steps that are taken when a message is signed by a sender and verified by a receiver:

1. The sender calculates the message's fingerprint.

2. He encrypts the digest with his private key resulting in the signature.

3. He forwards both original message and digital signature to the receiver.

4. The receiver decrypts the signature with the help of the sender's public key.

5. The receiver applies the same hash algorithm the sender used to the original message.

6. He compares the results of steps 4 and 5. If they are equal, the signature is verified. If not, verification failed.

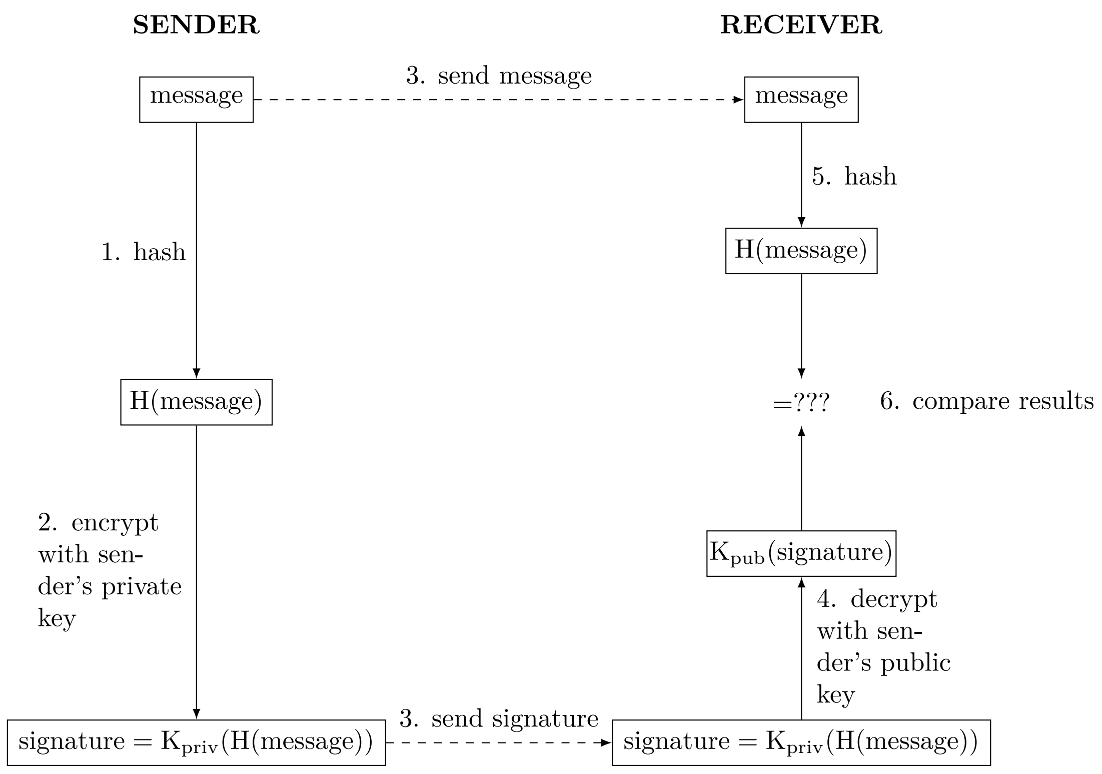

SENDER
RECEIVER
message
3. send message
message
5. hash
1. hash
H(message)
H(message)
=???
6. compare results
2. encrypt with sender's private key
K_{pub}(signature)
4. decrypt with sender's public key
signature = K_{priv}(H(message))
3. send signature
signature = K_{priv}(H(message))

Figure 2.8: Digital signature steps. Adapted from [Gri09], Chapter 7, page 19

2.4 Message Digest Algorithm MD5

MD5 is a message digest algorithm often used for checking the integrity of files. It was designed by Rivest in 1992 as an extension of MD4 [Riv92]. Today, it is no longer considered as collision-resistant, that means it is possible to find two texts M and M' with equal digest values. It is possible to use MD5 for authenticating routing updates in RIPv2, though default settings do only take cleartext authentication into consideration.

MD5 gets an input message of any length and computes so-called *message digests* or *fingerprints* as an output. It produces an output of 128 bits after processing four rounds, each of which consists of 16 steps. Each step depends on the output of the previous step.

A message M for which a digest is to be computed has to be prepared in a certain way. First, *padding bits* are appended to the message. Independent of Ms length, one bit 1 is added and then as many bits 0 so that the length of M is

$\equiv 448 \bmod 512$. Next, a 64 bit representation of Ms original length is attached to the output of the previous step. The outcome is divisible by 16 32 bit words M_i without remainder. In the following step a buffer is initialized to certain fixed constants. It consists of four 32 bit words A, B, C and D.

For the execution of the main algorithm four auxiliary functions (F-functions) are needed. They each get three 32 bit words as an input and compute an output of 32 bits. In each of the rounds another function is used. The functions are as follows:

$$\begin{aligned}
F(B,C,D) &= (B \wedge C) \vee (\neg B \wedge D)\\
G(B,C,D) &= (B \wedge D) \vee (C \wedge \neg D)\\
H(B,C,D) &= B \oplus C \oplus D\\
I(B,C,D) &= C \oplus (B \vee \neg D)
\end{aligned}$$

The original message is divided into 512 bit blocks, each block runs through four rounds of the following computations (see Figure 2.9):

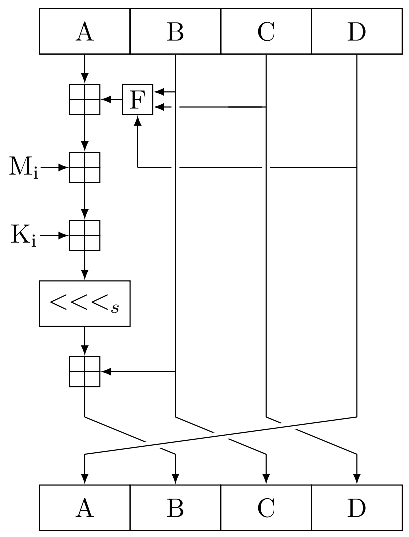


Figure 2.9: Main algorithm of MD5. Adapted from [ius11]

1. F (respectively G, H, and I in the later rounds) is computed with input B, C and D.

2. The output of F is added (denoted by $\boxplus$) to A. Here, “added’ means an addition modulo 2^{32}.

3. One of the M_is of the 512 bit block is added.
4. A constant K_i is added.
5. The output is left-rotated bitwise for s positions (denoted by $<<<_s$); s is different each time.
6. B is added.
7. In the next round the outcome of the previous computation becomes the new B, the old D becomes the new A, the old B becomes the new C and the old C becomes the new D.

Once a 512 bit block has run through four rounds, the next 512 bit block is processed. These steps are iterated until the whole message is processed. The concatenation of the values stored in A, B, C and D at this point represents the 128bit message digest of the original message.

3 Securing Distance Vector Routing Protocols

In this chapter, different approaches to securing distance vector routing protocols are presented. Since the second version of the routing information protocol, using an authentication algorithm is possible, yet it is not mandatory. The first section introduces RIPv2 with MD5 authentication. Section 2 gives a description of a solution that uses digital signatures, sequence numbers and a recursive path traversal mechanism. In Section 3, RIP-TP, an approach that uses a triangle theorem and probing messages to verify distances is shown. Next, PAIR, an algorithm that uses predecessor information and path sum metrics to detect attacks is described. Under certain conditions even recovery from malicious updates is possible. In Sections 5 and 6 the concepts of S-RIP and S-DV are explained. They both use signatures and consistency checks for validating updates, but S-RIP involves node reputations whereas S-DV introduces trusted routers. Finally, a symmetric key based approach is described, in which every node shares a set of keys with every other node in a network.

3.1 Authentication in RIPv2

In RIPv1, the early version of the routing information protocol, authentication was not provided. This matter of fact makes RIPv1 insecure. In RIPv2 however, extensible authentication mechanisms were added to the protocol; it offers the possibility of using either cleartext authentication or using hash algorithms – keyed MD5 or the SHA family – for securing the authentication process. This chapter explains authentication in RIPv2 by taking the example of MD5 (for further details on MD5 see Section 2.4). Authentication using a SHA hash algorithm proceeds similarly.

Without any authentication measures at all it is very easy for intruders to inject bogus routing updates and corrupt a network's routing topology. Only simple misconfigurations can be discovered. Even if cleartext authentication is used – as it is possible in RIPv2 – a deceiver can wiretap the password while it is sent over the network in the clear. As a consequence, cleartext authentication is no longer considered adequate and hash algorithms are brought into effect [AF07]. When using hash algorithms the output of the algorithm is transferred instead of the authentication key that should remain secret. Forged routing updates and attacks on the network are then more likely to be discovered. Neither cleartext authentication nor the use of hash algorithms offers confidentiality, as the content of the message stays in the clear. This is not considered a drawback because in routing protocols it is not important to keep updates secret.

When keyed MD5 authentication is used, the RIPv2 packet format described in Section 2.1.2 has to be altered as can be seen in Figure 3.1.

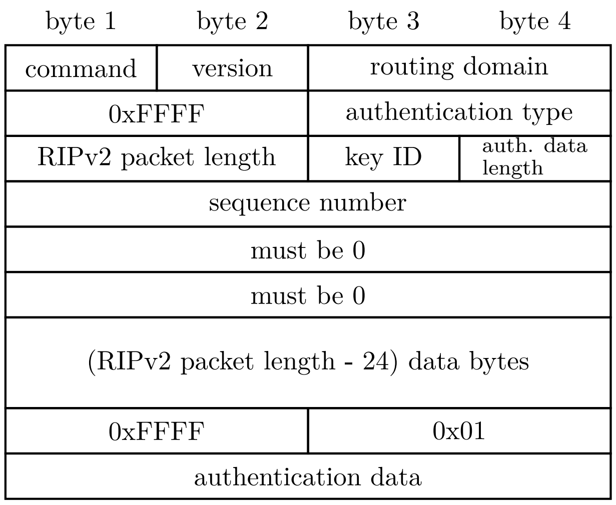


Figure 3.1: Format of a RIPv2 packet that uses MD5 authentication [AF07]

The header stays the same but the 16 byte authentication key field is replaced by five new fields that define a *keyed message digest* trailer:

- *authentication type*: Is set to 3 denoting *keyed MD5.*
- *RIPv2 packet length*: Indicates the length of the RIPv2 packet (without the authentication trailer).

- *key ID*: Helps to identify the key that is used to compute the authentication data for this particular packet. [AF07] makes use of a *Security Association* which is explained below.
- *authentication data length*: States the length of the trailing authentication data field. Due to this field other algorithms can be used instead of MD5.
- *sequence number*: The sequence number must be non-decreasing for all packets that have the same key ID.
- *authentication data*: This field in the trailer of the message contains the output of the hash function.

The Security Association mentioned above is an agreement between two network interfaces (sender and receiver) that helps them to secure their communication by sharing certain attributes. Along with other parameters, it includes the authentication key which is used to calculate the authentication data for this specific RIPv2 message. The sender selects the Security Association based on the outgoing router interface and fills out the *key ID* field accordingly [AF07]. When receiving a packet the receiver can tell from the key ID and the inbound interface which Security Association to use. At least, a Security Association must contain a key ID, an authentication key, the authentication algorithm that is in use, a 32 bit sequence number that must not decrease for a given key ID value and sender, a start time and a stop time.

When a RIPv2 packet is created, a Security Association is chosen according to the packet's outbound interface. Next, the fields that contain RIPv2 packet length, key ID and authentication data length are filled in so that they match the chosen Security Association. Once those fields are filled the message is processed according to the algorithm in use. For MD5, the following steps are taken:

- The 16 byte RIPv2 authentication key is appended to the message in memory.
- The message is prepared for being processed by adding padding bits (see Section 2.4) in memory.
- The message digest (= authentication data) is calculated using the MD5 algorithm (see Section 2.4).

After the hash algorithm has been executed the authentication data is written into the authentication data field and the message is transmitted.

The first step the receiver has to undertake upon reception of the message is to truncate the authentication data and set it aside. After that it determines the RIPv2 Security Association from the key ID and the incoming interface. In case it cannot determine a valid Security Association, it immediately stops processing the packet. Moreover, a security event should be recorded by the RIPv2 subsystem. If however a Security Association is found, the packet is processed according to the algorithm specified in the authentication type field (MD5 in this case) and the key to which the key ID field refers. The output of this step constitutes the authentication data. The receiver now compares the calculated authentication data with the authentication data it previously received and set aside. If they match, authentication was successful, the authentication trailer is cut off and the packet is treated like a regular RIPv2 packet. If calculated and received authentication data differ, the packet must not be accepted and a security event should be logged by the RIPv2 subsystem.

Using sequence numbers is also conducive to security. Normally, a newly received packet has a sequence number that is higher than the sequence number of the previously received message. If this is not the case, the packet must be discarded unprocessed and a security event should be recorded. Sometimes connectivity between two neighbors gets lost; the receiver should then accept packets with higher sequence numbers than the one it last received or packets with a sequence number of zero. Although this mechanism does not completely protect from replay attacks it makes them more unlikely.

Any mechanism that uses cryptographic keys has to oversee distribution and use of keys in order to prevent fraudulent use and problems that arise from mismanagement. Most important is to keep authentication keys secret and to never send them over the network in the clear. Keys should be changed regularly and the lifetime of a certain Security Association should be limited. To avoid problems in transition periods when keys are changed a system must support the storage and usage of more than one key on a certain interface at the same time. Thus, the end of the old Security Association's lifetime and the beginning of the new Security Association's lifetime can overlap and the system can use either Security Association (and hence either key) for authentication. Thanks to this so called *smooth key rollover* it is not necessary that all participants change their Security Associations at once and the risk of losing packets through invalid keys is banned.

Another aspect that needs to be considered is how routers that do not use the same RIP implementations handle authentication. Even for routers in the same network it is not mandatory to use authentication if other routers in that domain do so, and it is also not required that routers use the same version of the routing information protocol. Therefore, it should be specified in all possible cases how routers react to incoming routing updates [Mal94]. A router running RIPv1 that receives an authenticated message ignores it as the address identifier field contains *0xFFFF* and thus differs from the address family IP. A router that is not configured to authenticate RIPv2 messages accepts only RIPv1 and unauthenticated RIPv2 messages and should reject authenticated RIPv2 packets. If however a router is configured to authenticate RIPv2 packets, it should only accept RIPv1 messages and RIPv2 messages that can be authenticated. Messages that do not pass the authentication test or are unauthenticated should be rejected. Yet in order to increase security it would be best to reject RIPv1 messages if authentication is active [Mal94].

3.2 Implicit Path Method Using Predecessor Information

In his master's thesis [Smi97], Smith proposes several mechanisms to secure distance vector routing protocols. When they are used in combination they provide effective protection against the exploitation of certain known vulnerabilities. For example, unauthenticated RIPv2 assumes that neighboring routers trust each other and even that all routers in a network trust each other transitively. Smith's solution only requires that routers trust the information they get about links that are directly connected to them. In contrast to earlier solutions, where only the next hop on a route to the destination can be validated, this approach additionally confirms that the next hop leads to the destination at the cost that is listed in the routing update.

Smith identifies four classes of threats: Disclosure, deception, disruption and usurpation. Countermeasures provide security services (confidentiality, integrity, authenticity, access control, non-repudiation, availability) that in turn help to fight possible threats (for details see Section 2.2). In a successful attack, the invader can choose to treat routing information according to his malicious intentions. This can be invention or alteration of routing updates if he aims to change the network's topology; he can also replay or discard updates in order to restore a previous topology. Thus, mechanisms are needed that provide access control, authentication and integrity. By introducing sequence numbers or time stamps for routing updates, using digital signatures and adding predecessor information as well as destination link costs to routing updates, the author offers a solution to the above mentioned problems.

According to [Smi97] there are two different types of communication in distance vector routing protocols:

1. Communication between neighbors: Aggregated routing updates that are specifically useful for the destined receiver are sent from one neighbor to another neighbor.

2. Communication between remote routers: A router can send fields of routing updates that define a certain destination to any group of remote routers that is designated by routing decisions (dynamic multicast).

In the following, solutions for both types of communication are presented. For the functioning of these measures it is assumed that every router holds a public key pair and that it can rely on routing information it gets from other routers only regarding directly connected links.

Protecting routing messages: Integrity and authenticity of routing messages can be ensured by using sequence numbers and digital signatures. A router that joins a network assigns a sequence number of zero to the first routing message it sends out. The sequence numbers of the following messages are increased by one for each message that is sent out. If another router receives a routing message with a sequence number that it already has seen, or receives a message with a sequence number higher than the one it expected next, this indicates a violation and the session is reset. By this means, replay and deletion of messages can be prevented. In addition, the originating router digitally signs each message to ensure authenticity and integrity. The signature helps to reveal manipulations of messages; they should be discarded if an inconsistency is found.

Protecting routing updates: Not only routing messages are provided with sequence numbers to detect replay and deletion, sequence information is also added to each single update. When new routes are computed, each neighbor of the router that generated the new route gets an update. The updates all carry the same sequence number (a time stamp is also possible), as it is possible that a router that is multiple hops away from the originating router receives the same route in several different updates. This is because the updates all take different paths but represent the same destination. If they had different sequence numbers, only the update with the highest one would be taken into consideration although they all should be taken into account. Thus, they get the same sequence information to solve the problem.

Integrity of a route and the corresponding distance can be verified if predecessor information is added to a routing update. A router that receives an update and wants to verify integrity and authenticity of a route that is contained, has to ensure that the reported distance matches a path which is valid and authentic all along the way and that the route starts with the next hop (successor) that is listed in the update. Adding predecessor information (meaning the second to last hop on a route to the destination) to an update makes it possible to deduce implicit paths, which means that first routers adjacent to the destination are located and then intermediate routers that are again adjacent to them and so on, until the first router on the path is reached. An additional method to validate a route is to verify the distance that is indicated in the routing update. This can be accomplished by adding the destination link cost to the update so that during path traversal (as described above) the link costs of the implicit path can be summed up. If the computed sum equals the advertised distance, this validation step is successful.

The use of digital signatures helps to protect the sequence number, destination, predecessor and destination link cost fields in a routing update from fabrication or alteration by subverted routers. The originating router signs all these fields, but not the distance field, and adds the signature to the routing update which it then sends to its neighbors. A router that receives the signed update can then authenticate these fields and validate the distance field as previously explained. It is important to cross-check destination and predecessor of an advertised route so that manipulation of both fields is averted. In RIPv2, a possible way for cross-checking assumes that a node can be connected to a network if the network address and the mask of the network, as well as the node's corresponding IP address, are known. Furthermore, there must be a trusted network authority that administers IP addresses to node names and node names to public key mappings [Smi97]. When a router now advertises an update, it does not only equip it with the usual fields but also adds the IP addresses of its interfaces to the destination and predecessor networks and signs them along with the other fields. The receiver checks if the interface addresses it learns from the update are part of the associated network. If so, the addresses further have to represent the same public key which validates the signature. In case all conditions are met it is assumed that the predecessor and destination information is authentic and unaltered. In interdomain routing protocols there are other ways for cross-checking destination and predecessor, but this is not further explained here.

Cost of countermeasures: Figure 3.2 shows the relation between routing messages and routing updates and illustrates what kind of information is added to each of them using the approach described here.

Naturally, the security measures presented here entail certain costs in space and time. New information is added to routing messages and routing updates, and the newly added information has to be processed. As a result, a routing message grows by 68 bytes; 64 bytes are taken up by the digital signature, 4 bytes are needed for storing the sequence number. The growth per routing update is even higher, each update grows by 85 bytes. The sequence number (or time stamp) requires 4 bytes, the destination link cost requires 1 byte, the digital signature needs 64 bytes of storage space and if IPv4 IP addresses are assumed, predecessor network address, predecessor network mask, IP address of the router's interface on the predecessor network and IP address on the router's interface on the destination network each require 4 bytes.

The expenditure of time per routing message that emerges from the countermeasures equals the amount of time that the originating router needs for calcu-

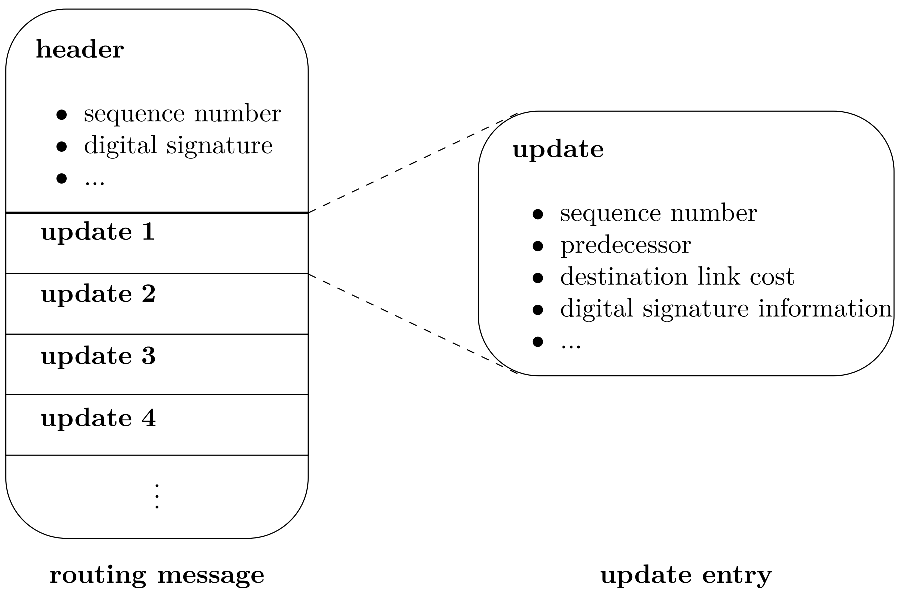


Figure 3.2: Additional information in routing message and update

lating digital signature and sequence number and the receiving router needs for their verification. Per update, a digital signature is calculated for each link of the router that generates the update, as the predecessor is different for each interface. In addition, the signature has to be verified when a router uses a route that contains the link from the update in its implied path. According to [GLAM97] the overhead caused by path traversals when a new route to a destination is selected can be almost neglected when efficient algorithms are used.

Pseudocode: Smith sums up his set of countermeasures in pseudocode which is adopted one-to-one in the following, as it vividly demonstrates the operating principles. First, a number of data structures are defined (source: [Smi97], page 38):

- **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, received by router i from neighboring router k is a column vector of update entries with the following fields:
  - j: Destination
  - UD_j^k: Distance from k to j
  - up_j^k: Predecessor network
  - usn_j^k: Update sequence number
  - ul_j^k: Link cost of j
  - uds_j^k: 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.
- **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_{jk}^i: Distance from k to j
  - p_{jk}^i: Predecessor network
  - sn_{jk}^i: Update sequence number
  - l_{jk}^i: Link cost of j
  - ds_{jk}^i: 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.

- **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.
  - $RTmod^i_j$: A boolean indicating whether this entry has been modified.

In addition, a number of routines are called in the pseudocode, 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.

These structures and routines apply in the pseudocode:

Algorithm 3.1: Secure distance vector processing according to Smith [Smi97]

```
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 $(j, UD^k_j, up^k_j, usn^k_j, ul^k_j, uds^k_j)$ in $U_k$ do
    begin
        $o \leftarrow$ Originator($ds$);
        if (($usn^k_j \geq SN^i_{jo}$) and ($ds =$ DigSig($j, p, sn, l, o$)))
        then begin
            $D^i_{jk} \leftarrow UD^k_j$;   $p^i_{jk} \leftarrow up^k_j$;   $sn^i_{jk} \leftarrow usn^k_j$;
            $l^i_{jk} \leftarrow ul^k_j$;   $ds^i_{jk} \leftarrow uds^k_j$;   $SN^i_{jo} \leftarrow usn^k_j$;
        end
    end
    call UpdateRT($i$);
end

function ValidatePath($i, k, d, p, l$) $\longrightarrow$ boolean;
begin
    $tj \leftarrow p$;   $p \leftarrow p^i_{tj,k}$;   $td \leftarrow l$;
    while (($p$ not in $L_i$) and ($p \neq$ null)) do
    begin
        $td \leftarrow td + l^i_{tj,k}$;   $tj \leftarrow p$;   $p \leftarrow p^i_{tj,k}$;
    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
            $(D^i_{jx}, p^i_{jx}, sn^i_{jx}, l^i_{jx}, ds^i_{jx}) \leftarrow$ SelectRoute($i, j$);
        until (($x \neq$ null) or ValidatePath($i, x, D^i_{jx}, p^i_{jx}, l^i_{jx}$));
        if (($x \neq$ null) and (($D^i_j \neq D^i_{jx}$) or ($s^i_j \neq x$) or ($ds^i_j \neq ds^i_{jx}$)))
        then begin
            $D^i_j \leftarrow D^i_{jx} + l_{Network(x)}$;   $s^i_j \leftarrow x$;   $p^i_j \leftarrow p^i_{jx}$;
            $sn^i_j \leftarrow sn^i_{jx}$;   $ds^i_j \leftarrow ds^i_{jx}$;
            $RTmod^i_j \leftarrow$ 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 $RTmod^i_j =$ true do
    begin
        $U_{tmp} \leftarrow U_{tmp} \cup (j, D^i_j, p^i_j, sn^i_j, l^i_j, ds^i_j)$;
        $RTmod^i_j \leftarrow$ 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
```

Efficiency analysis: The use of digital signatures for routing messages is the measure that offers protection against the majority of attacker/attack combinations, i.e. protection against fabrication and modification by unauthorized routers, masquerading routers and subverted links (cf. Section 2.2). Assigning sequence numbers further protects against replay and deletion by unauthorized or masquerading routers and subverted links. In brief, the countermeasures prevent messages from being manipulated by unauthorized routers that do not know the cryptographic keys.

Subverted routers however *do* know the cryptographic keys and thus could compromise the routing procedure. This is where update protection countermeasures are brought into effect. Signed updates cannot be invented or modified by subverted routers and, as in the case of routing messages, sequence numbers or time stamps defeat replay of updates. If an update also carries information on the predecessor network, the receiver can calculate and verify the implicit path and successor information. If, on top of that, it contains the destination link cost, the distance can also be verified.

Yet, the presented countermeasures cannot safeguard from all kinds of threats. If one acts on the assumption that a router always provides correct information about connected links, it is possible that a subverted originating router manipulates destination link costs. As long as each link has the same cost attached to it (which is the case when hop count is used) there is no risk involved. But as soon as different link costs are possible, risk rises with rising link costs. Also, there is no way to efficiently prevent attackers from disclosing routing information. Only encryption could solve this problem, but as this would mean that every neighbor of the originating router would get its own encrypted copy of the update this would result in an unreasonably high amount of overhead.

3.3 RIP-TP: RIP with Triangle Theorem Checking and Probing

RIP with Triangle Theorem Checking and Probing (RIP-TP) is a routing update validation algorithm developed by Pei, Massey and Zhang. The algorithm helps to detect questionable updates and sends probing messages to certain nodes in order to validate the distances that have been announced. In distance vector routing protocols a node cannot be sure if the information it holds regarding the distance to nodes that are not its direct neighbors is correct. The authors bring in an example that has happened in the early ARPANET [PMZ03]: A router located on the US east coast advertised a route with a distance of zero to UCLA, located on the US west coast. This false route was caused by a memory fault, other routers believed it to be true and sent their traffic destined for UCLA via the faulty router which then dropped the packets. According to Pei, Massey and Zhang, this *black hole* could have been avoided if RIP-TP had been implemented in the ARPANET. A router can use RIP-TP and profit from its abilities even if no other router has implemented RIP-TP. Other routers merely must be able to respond to UDP probing messages, but this is given, as it is part of ICMP functionality. Moreover, RIP-TP can be used in RIPv2.

As it is very common in RIP, RIP-TP also uses hop-count as the distance metric. The triangle theorem states that after the routing protocol has converged, for any subset of three nodes the following condition holds: The distance between one pair of the three nodes must be equal or less than the sum of the distances of the other two pairs, or

$$dist(A, C) \leq dist(A, B) + dist(B, C)$$

for any three nodes A, B and C [PMZ03]. In standard RIPv2 a node accepts new distances which it gets in a routing update without validating them. This is different in RIP-TP. Figure 3.3 illustrates how RIP-TP's triangle checking works. Node A and node B are directly connected (depicted by the drawn-through line), nodes A and C and nodes B and C do not have a direct connection (depicted by the dotted lines). If node B advertises to A $dist(B, C)$, node A checks whether $dist(A, C) \leq dist(A, B) + dist(B, C)$ holds. This is check 1 of

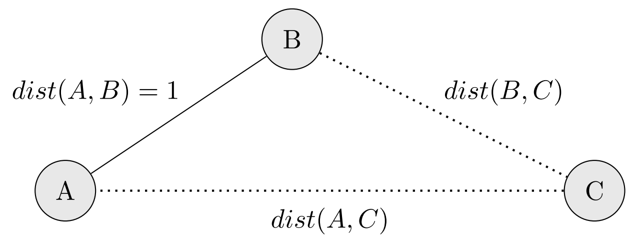


Figure 3.3: Check 1 in a network running RIP-TP

the triangle theorem; if it does not hold true for the newly received routing update, an inconsistency is detected. Yet, it cannot be stated which of the distances created the inconsistency. The only certain fact is that $dist(A, B) = 1$ as they are neighbors and A received the update from B which leaves $dist(A, C)$ or $dist(B, C)$ as possible candidates for causing inconsistency. Note that the violation could have been caused by a false distance that has been injected or by routing updates that are delayed or got lost.

Now the help of a second check is used to identify if either $dist(A, C)$ or $dist(B, C)$ is faulty. The advertised and possibly incorrect distance $dist(B, C)$ will not be accepted without further checking and is marked as potentially invalid. Assuming the next hop from node A to node C is node D ($nexthop(A, C) = D$)), see Figure 3.4, node A checks if $dist(D, C) \leq dist(D, B) + dist(B, C)$. Node A knows that $dist(D, B) \leq 2$ as both nodes are its neighbors. An advertised

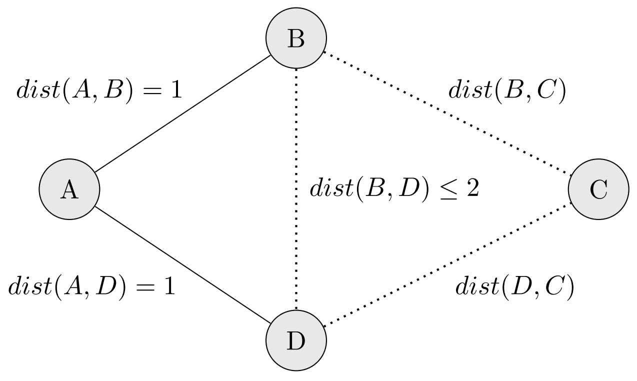


Figure 3.4: Check 2 in a network running RIP-TP

distance that passes check 1 is not checked any further. Thus, newly learned distances belong to one of the following groups:

- Distances that pass check 1: They will not be examined closer and are accepted.
- Distances that fail *only* check 1: They are potentially invalid.
- Distances that fail check 1 *and* check 2: They are probably invalid.

Suspicious distances can be either potentially or probably invalid and thus are subject to closer examination. Under ideal conditions all questionable distances could be verified; however in practice, this would produce far too much verification overhead. So it is important to first validate the distances that are more likely to be invalid; this is done by sending probing messages. Also the number of probing messages per received update should be limited, as in the worst case 25 RIP entries (cf. Section 2.1.2) could be potentially or probably invalid. In order to achieve this, a new threshold Z is introduced, which represents the maximum number of probing messages that can be sent per received update:

$$Z = max(Z_{min}, min(0.5 * X, Z_{max}))$$

X stands for the number of potentially invalid distances, Z_{min} and Z_{max} are control parameters, the authors use the values 2 and 5, respectively. Using this equation, in an update with 8 potentially invalid distances 4 probing messages would be allowed.

After both checks have been performed and the number of possible probing messages per update has been determined, UDP packets, which constitute said probing messages, are sent out. In the previous example, $dist(B, C)$ is questionable and node A wants to verify it. For this purpose it sends a UDP packet to node C, sets the packet's time to live to $dist(B, C) + 1$ and the destination port number to a UDP port number that is unused. Moreover, A starts a timer $(6 * (dist(B, C) + 1) * hop\_delay)$, where hop_delay is either measured by node A or configured. Several different scenarios are now possible:

- Node A receives an ICMP "destination unreachable" message sent by node C, the reason being "unreachable port". This shows that the probing message has actually arrived at its destination. Thus, C's answer is considered an acknowledgement and $dist(B, C)$ is verified.

- Node A receives an ICMP “time exceeded” message. It knows that the probing message was dropped due to time to live exhaustion and concludes that the actual distance from A to C is greater than $dist(B, C) + 1$.
- A node on the route between A and C does not know how to reach C. It drops the probing message and sends an ICMP “destination unreachable” message to node A.
- The network is congested or other problems occur. Then the probing message or the ICMP response might get lost and the timer expires.

The last three scenarios are considered negative acknowledgements which indicate that verification has failed.

Depending on the results of the verification procedure distances are handled differently. If a distance passes the verification test, the receiver incorporates it into its routing table. If it, however, cannot be verified, it is not accepted. There is a risk of valid distances being discarded in case probing messages or ICMP reply messages are delayed or get lost. Nevertheless, this is not a serious problem as in RIPv2, routers send out their routing updates every 30 seconds and a previously mistakenly rejected message is again subject to verification.

The algorithm is enhanced by introducing a verification bit which is set after a new distance has been successfully validated. Per distance, this means an overhead of only one bit, and the drawbacks brought about by using threshold Z, as mentioned above, can be overcome. Due to this threshold not all distances might run through the validation procedure if the number of potentially or probably invalid distances in a certain update was very high. Thus, invalid distances might remain unnoticed as the router only chooses up to Z potentially or probably invalid distances that undergo further checking and the rest is not checked. Wrong distances might be accepted and once they are installed in a router’s routing table they might “poison” it in such a way that future triangle checks produce wrong results. In order to prevent this, the verification bit is brought into effect. If indeed an invalid distance passes unnoticed, its validation bit is not set. As soon as a new periodic RIPv2 update arrives which contains the very same distance, this distance is again among the distances that need to be verified. If now the receiving router eventually receives an update with less than

Z potentially or probably invalid distances, it can choose up to Z - (*# probably invalid distances + # potentially invalid distances*) from its routing table that do not yet have their verification bits set, and verify them finally. Sooner or later every distance will be checked.

Another optimization is the introduction of a second threshold. When there are too many probably invalid distances in an update the whole update message is rejected, even if it contains distances that are valid. This is without detriment as a new update will arrive soon. On the other hand the update should not be completely discarded if it contains only few probably invalid distances. The new threshold helps to find the right balance:

$$thresh\_drop = max(thresh_{min}, min(0.5 * max(Z, S), thresh_{max}))$$

Here, S represents the number of probably invalid distances, $thresh_{min}$ and $thresh_{max}$ are control parameters (in [PMZ03], $thresh_{min} = 2$ and $thresh_{max} = 5$ are used). Now, the update message is only rejected if the amount of distances that did not pass validation exceeds $thresh\_drop$.

RIP-TP takes advantage of the fact that RIPv2 packets have unused fields available. It uses a 16 bit field to store a number that tells the sender's neighboring routers how many routes in that update have already been validated, counting from the first entry.

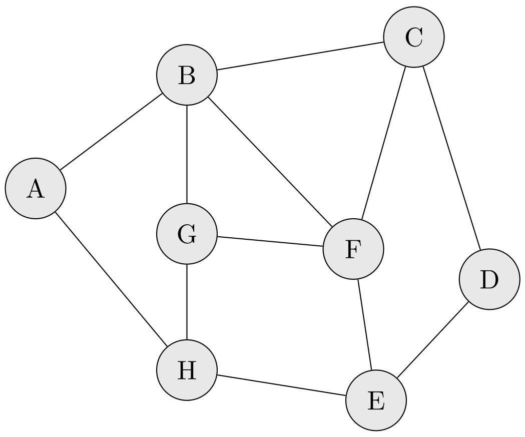
3.4 PAIR: A Pivot Based Algorithm for Inconsistency Recovery

Chakrabarti and Manimaran developed an algorithm that cannot only detect false routing updates under certain conditions but can also restore order after malicious updates have been received. They named their algorithm *Pivot Based Algorithm for Inconsistency Recovery* (PAIR) [CM03] as network nodes are categorized into different groups, one of them being called *pivots*. The approach focuses on router attacks, in which a malevolent router is either the source of a forged routing update or changes routing updates when forwarding them. Link attacks, in which routing messages are altered fraudulently by an intruder who gets access to a network link, are not considered in this solution.

For a better understanding of the algorithm a few definitions are useful:

- *transmitting node*: The node that sends the routing update.
- *receiving node*: The node that receives the routing update.
- *distance vector tree (τ)*: Upon reception of a routing update the receiving node uses predecessor information to build a tree τ whose root is the transmitting node.
- *pivot*: The pivot of two nodes i and j in τ is their lowest common ancestor on their path to the distance vector tree's root.
- *descendant (Δ)*: The descendants Δ_i of a node i in τ are the children of i.
- *predecessor (ρ)*: j is predecessor ρ_i of a node i in τ, if i is a descendant of j:
$$\rho_i = j \Leftrightarrow i \in \Delta_j$$
- *hop length (η)*: The hop length η_i of a node i in τ is the number of hops in the shortest path from the root node in τ to i.
- *path sum (σ)*: σ_i of a node i in τ is the sum of all path lengths of all paths that either pass i or end in i:
$$\sigma_i = \eta_i + \sum_{\forall j \in \Delta_i} \sigma_j$$
- *net path sum (θ)*: θ_i of a node i in a distance vector tree indicates the difference between σ that the receiver calculated and σ that the receiver received in the update. Let σ_i^r be the path sum that the receiving node got in an update, then:
$$\theta_i = \sigma_i - \sigma_i^r$$

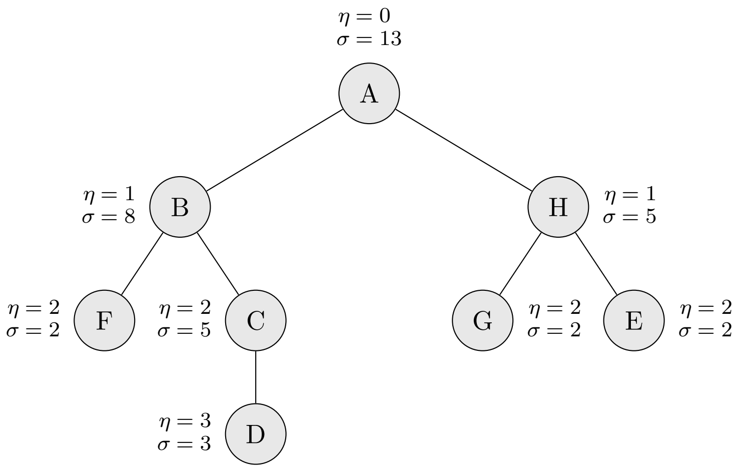
Figure 3.5 shows an example network, a table containing the information that node A sends to its neighbors and the corresponding distance vector tree that A's neighbors can build up from the received information. Subfigure (c) also shows hop lengths and path sums for each node in the network. In this example, the calculated path sum values do not differ from the received values which means that the update was not altered and is valid.



(a) Example network in PAIR

id	ρ	σ
A	A	13
B	A	8
C	B	5
D	C	3
E	H	2
F	B	2
G	H	2
H	A	5

(b) Information that A sends to its neighbors



(c) Tree constructed from A's update information

Figure 3.5: Example for tree construction in PAIR

According to [CM03], PAIR consists of four phases which are explained in detail in the following paragraphs. First, the node that receives a new update builds up the distance vector tree with the help of the predecessor information from the update. Then, hop lengths, predecessors and path sums of each node are calculated using the newly built tree. Next, faulty updates can be discovered by comparing received and calculated values. Finally, inconsistency recovery can be performed if malicious information has been detected.

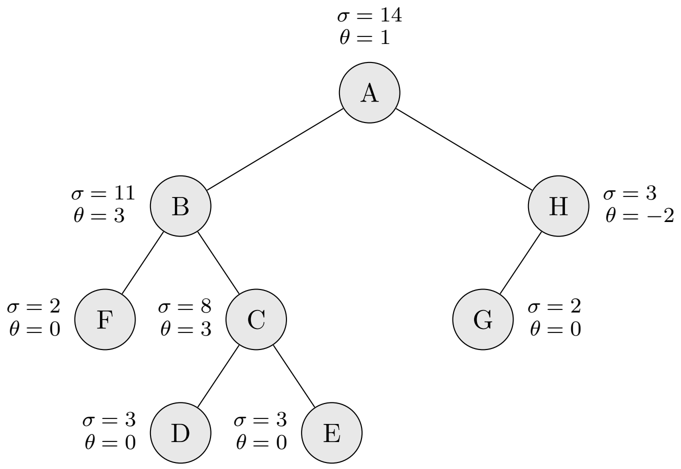
Tree construction: The node from which the update was received constitutes the root of the distance vector tree. In our example node A is the root. Then the receiver uses the help of the table's predecessor information to determine which nodes the root node is the predecessor of. These nodes (in our example node B and node H) constitute the children of the root and are added below and connected with their predecessor. For each newly added child the receiver again looks up which node they are the predecessors of, adds these nodes as their children and so on. This continues until no more nodes can be added to the tree.

Updating of metrics: The newly constructed distance vector tree is used to calculate hop length η, path sum σ and net path sum θ for each node in the tree according to the equations given above.

Detection procedure: PAIR is able to discover forged routing updates if not more than one predecessor/path sum entry per distance vector tree has been altered. If for any node this information has been manipulated, some θ in the tree are $\neq 0$. Figure 3.6 shows what the distance vector tree looks like after an attacker has changed the routing entry for node E. It wants to make the receiver believe that E's predecessor is C instead of H. As can be seen from Figure 3.6(b) the receiver is mislead and actually constructs a distance vector tree that does not match the reality shown in Figure 3.5(c). But as soon as it calculates path sums σ and net path sums θ for each node in the network, it detects that nodes A, B, C and H have non-zero θ values and thus concludes that the routing update has been manipulated.

id	ρ	σ
A	A	13
B	A	8
C	B	5
D	C	3
E	C	3
F	B	2
G	H	2
H	A	5

(a) Update with forged entry for node *E*



(b) Tree constructed from manipulated update

Figure 3.6: Manipulation detection in PAIR

Recovery procedure: After a bogus distance vector update has been received PAIR is capable of recovering if the following conditions are fulfilled:

- The received predecessor information does not contain loops, that means $\sigma_i > 0 \quad \forall i \in \tau$.
- The root node's net path sum must not be 0 ($\theta_x \neq 0$ if x is the root node). That means the hop lengths of the actual and the changed predecessor must differ.
- Either σ *or* ρ for only a single node in the network has been manipulated.

If these conditions are complied with, PAIR next classifies the network nodes into groups. Figure 3.7 illustrates the distance vector trees for the network shown in Figure 3.5 after the receiving node received different updates in which the predecessor information for one node has been changed. Figure 3.7(a) shows the unaltered topology. In Figure 3.7(b) *E*'s predecessor has been changed from *H* to *C*, in Figure 3.7(c) *D*'s predecessor has been changed from *C* to *B* and in Figure 3.7(d) *F*'s predecessor has been changed from *B* to *D*.

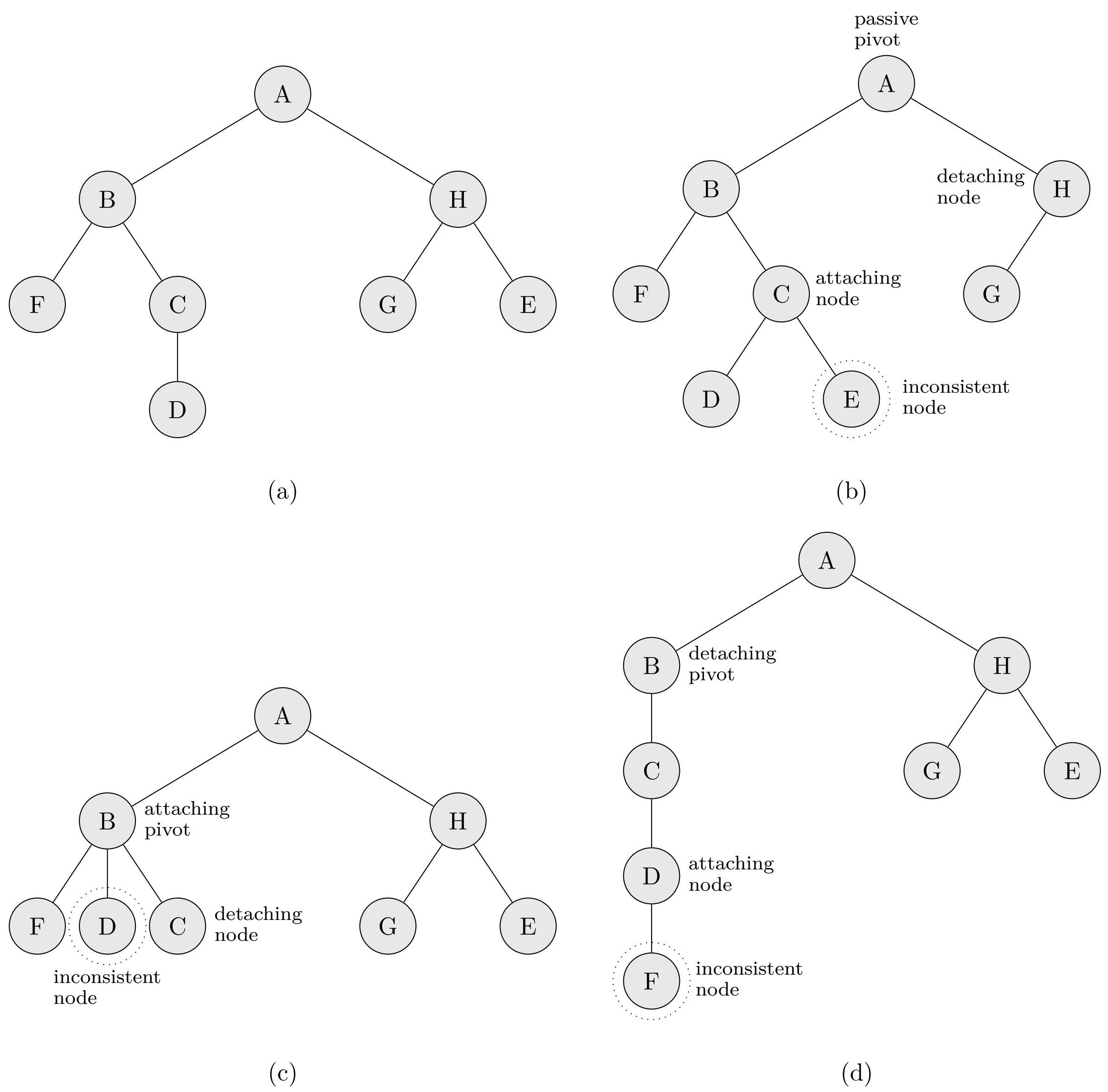


Figure 3.7: Classification of nodes in PAIR, adapted from [CM03], page 3

A node with changed predecessor is called *inconsistent node*. In Figure 3.7 inconsistent nodes are marked by a dotted circle. The actual predecessor – the node from which the inconsistent node has been delinked – is called *detaching node*. In Figures 3.7(b), (c), and (d) the detaching nodes are *H*, *C* and *B* respectively. The calculated predecessor – the node to which the inconsistent node has been linked – is called *attaching node*. In our example these are nodes *C* (b), B (c) and D (d). The pivot of the detaching and attaching nodes is called a *detaching pivot* if it is a detaching node by itself, it is called an *attaching pivot* if it is an attaching node by itself and it is called a *passive pivot* if it is neither a detaching nor an attaching node. In Figure 3.7(b) node A is a passive pivot, in (c) node B is an attaching pivot and in (d) node B is a detaching pivot. Nodes that do not fit into any of these categories are called *common nodes*.

Now that the possible kinds of nodes have been named it still has to be elucidated how to detect which one is which. This task is supported by seven properties [CM03]:

1. Property: If x is a **common node** and $\theta_x \neq 0$, then there is only one node x_i such that $\theta_{x_i} = \theta_x$ and $\theta_{x_j} = 0 \quad \forall j \in \Delta_x, i \neq j$.

2. Property: If p is the pivot and λ is the **inconsistent node**, then $\theta_p = \theta_\lambda$.

3. Property: If a node p is a **passive pivot**, then there are exactly two nodes among the descendant nodes of p (p^+ and p^-) such that $\theta_p = \theta_{p^+} + \theta_{p^-}$, where $\theta_{p^+} > 0, \theta_{p^-} < 0$.

4. Property: If node p is a **detaching pivot**, then there is exactly one descendant x of p such that $\theta_x > \theta_p > 0$.

5. Property: If node p is an **attaching pivot**, then there are exactly two descendants of p (x and y) such that $\theta_x < \theta_y = \theta_p < 0$.

6. Property: If x is a **non-pivot detaching node**, then $\theta_x < 0$ and $\theta_y = 0 \quad \forall y \in \Delta_x$.

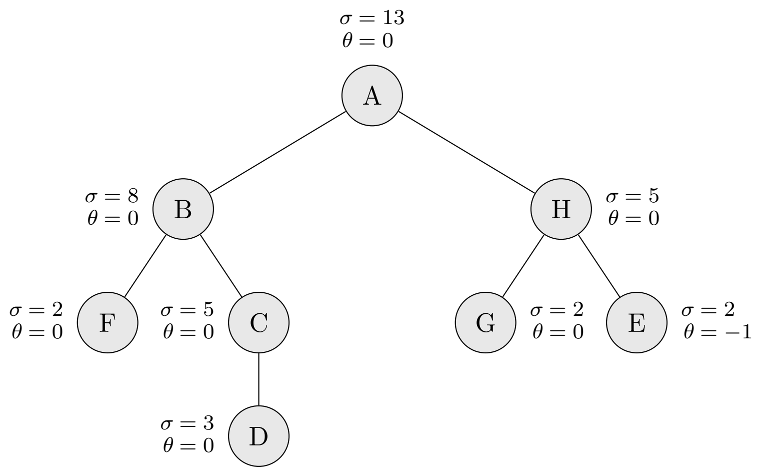
7. Property: If x is a **non-pivot attaching node**, then there exists exactly one descendant y of x such that $\theta_y = \theta_p$ where p is the pivot.

In order to determine how to recover from a malicious update the kind of inconsistency has to be identified. If only one net path sum value θ in the tree constructed by the update's receiver is different from zero, this is referred to as *σ-inconsistency*. If, however, more than one node has non-zero θ values, this is called *ρ-inconsistency*. Compared to recovery from ρ-inconsistency, recovery from σ-inconsistency is relatively easy. The receiver of the update knows which node's

σ value has been altered (the node with $\theta \neq 0$) and just takes the calculated σ value as the correct one. Figure 3.8 shows an example for σ-inconsistency. The only value that has been altered is the σ value of node *E*. The receiver constructs the tree as shown in the figure and calculates path sum and net path sum values for all nodes. As only one θ value differs from 0, the receiver concludes that σ-inconsistency exists. It now just accepts the calculated value as the actual one for node *E*.

id	ρ	σ
A	A	13
B	A	8
C	B	5
D	C	3
E	H	3
F	B	2
G	H	2
H	A	5

(a) Update with manipulated σ values for node *E*



(b) Tree constructed from manipulated update

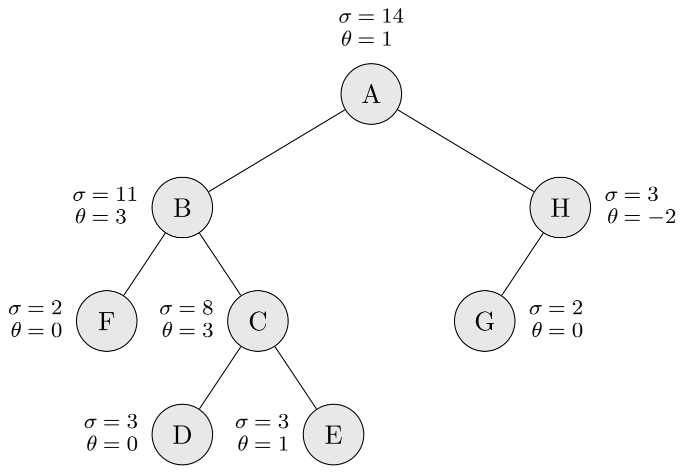
Figure 3.8: Recovery from σ-inconsistency in PAIR

Recovery in case of ρ-inconsistency is more complex and requires more steps. First, the pivot is located with the help of properties 1, 3, 4 and 5. Properties 3, 4 and 5 also help to determine if the pivot is a passive pivot, a detaching pivot or an attaching pivot, respectively. The inconsistent node is found with the help of property 2, the attaching and detaching nodes are located with the aid of properties 6 and 7. Once all nodes are classified the distance vector tree can be assembled properly: The inconsistent node is delinked from the attaching node because it “does not belong there” and the detaching node is made the predecessor of the inconsistent node by linking it to the detaching node. To check if the update is consistent after reassembling the distance vector tree, the net path sums of all nodes are recalculated. Recovery was successful if they are now all zero.

An example for recovery in case of ρ-inconsistency is given in Figure 3.9. The

id	ρ	σ
A	A	13
B	A	8
C	B	5
D	C	3
E	C	2
F	B	2
G	H	2
H	A	5

(a) Update with forged ρ entry for node E



(b) Tree constructed from manipulated update

Figure 3.9: Recovery from ρ-inconsistency in PAIR

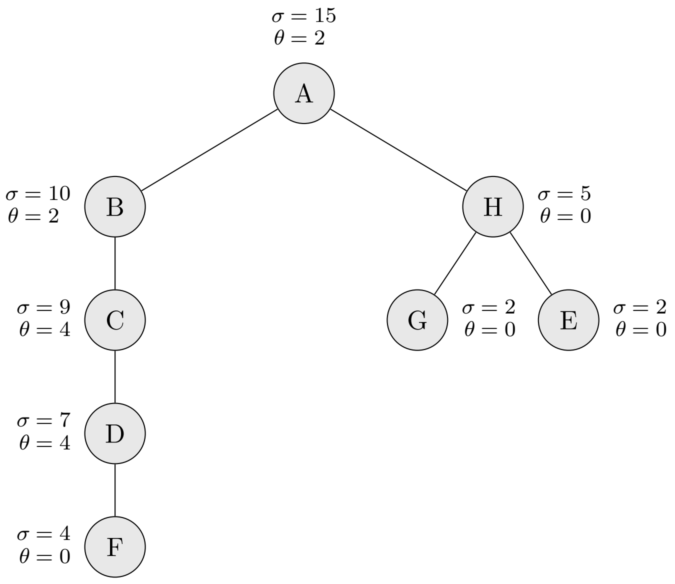
predecessor information for node E has been changed from H to C, whereas the path sum entry remained unaltered (recall that *either* ρ *or* σ for only *one* entry in the whole update may be changed in order to make recovery possible). Upon receipt of the new (forged) distance vector update the receiving nodes construct the distance vector tree shown in Figure 3.9(b) from the predecessor information in Table 3.9(a) without taking into account the path sum values given in the table. Next, they use the tree to compute path sums and net path sums (the differences between calculated and received path sums) for all nodes. As there are nodes whose θ values are $\neq 0$, they can conclude that there is an inconsistency.

Using property 3, node A can be identified as the passive pivot. There are exactly two children of A (namely B and H), whose sum of θ values amounts to A's θ value, one of them being < 0 and the other being > 0. With the help of property 6, node H can be identified as the detaching node: Its θ value is < 0 and all its descendants' (in this case node G) θ values are 0. Property 7 helps to reveal the attaching node. Node C is the attaching node, as there is exactly one descendant of C whose θ value is equal to the pivot's θ value (θ of nodes E and A both are 1). The last node left to determine is the inconsistent node. We have to find a node whose θ value equals the pivot's θ value (property 2). This applies to node E. Now that all nodes have been identified, the inconsistent node E can be delinked from the attaching node C and can be linked to the detaching node H. After recovery the net path sums θ are 0 for all nodes, which shows that recovery was successful and the update became consistent.

Figure 3.10 illustrates that recovery is not possible in case more than one value is manipulated. Assume that in the update belonging to the network shown in Figure 3.5(a) and (c), predecessor *and* path sum information have been changed according to Figure 3.10(a). The receiving node uses the predecessor information to construct the distance vector tree (Figure 3.10(b)).

id	ρ	σ
A	A	13
B	A	8
C	B	5
D	C	3
E	H	2
F	D	4
G	H	2
H	A	5

(a) Update with manipulated ρ and σ values for node F



(b) Tree constructed from manipulated update

Figure 3.10: Example for impossible recovery in PAIR

One pair of entries (ρ and σ for node F) has been changed, so PAIR can still identify the update as malicious, because there are θ values different from 0. By comparing the original distance vector tree in Figure 3.5(c) to the distance vector tree constructed from the manipulated update in Figure 3.10(c) one can easily see that node F is the inconsistent node, node D is the non-pivot attaching node and node B is the detaching pivot. However, PAIR cannot identify all these nodes by using the seven properties and is unable to restore consistency. At least the detaching pivot can be located using property 4, as there is exactly one descendant of B, namely C, whose θ value is greater than B's θ value, and both values are greater than 0. Property 7 does not help to identify the non-pivot attaching node D, as there is no descendant of D with a θ value equal to that of the pivot. Further, node F cannot be identified as the inconsistent node with the help of property 2, because its θ value is not equal to the pivot's θ value.

3.5 Secure Distance Vector Routing Protocol S-RIP

S-Rip is an approach by Wan, Kranakis and van Oorschot that helps to secure distance vector routing protocols [WKV04]. Their objective is to solve the main problems in distance vector protocols which are – according to them – the absence of efficient authentication and authorization mechanisms as well as the missing possibility of validating routing updates. These two weaknesses make it easy for intruders to manipulate routing messages. Even if authentication mechanisms are used, only the origin of a routing update and the fact that it has not been altered on its way from sender to receiver can be confirmed. The correctness of an update's content cannot be validated.

S-RIP is based on RIP with MD5 authentication and adds certain features: Routers check the consistency of a newly learned route by consulting the routers they have learned that route from; the new route is only added to the routing table after it has been validated for its correctness. As it is not an easy task to determine if a route is correct, S-RIP considers a route to be correct if it is consistent among the nodes that have circulated it. Depending on which factor more importance is attached to – security or low overhead – the number of nodes to be consulted varies. A so-called *reputation-based framework* helps to quantify the routers that are needed. False routes will be discovered if the routers adjacent to the router that produced the false information are not malicious themselves and do not collaborate with the misbehaving node; the neighboring routers can constrain dissemination of incorrect routing updates. In fact, S-RIP does not ensure that a route that has been validated is ideal. It just verifies that routes are consistent, not taking into consideration the associated costs.

The main idea of S-RIP is the introduction of a *node reputation*, a numeric value that replaces the conception of trust and expresses how much other nodes can rely on the fact that this particular node will propagate correct routing updates in the future. A node's reputation can change in the course of time, depending on the node's behavior. The value is used for determining how many other nodes to consult in a consistency check, which is explained in detail below. In addition to calculating node reputation, S-RIP makes use of several other assumptions that help to prevent possible manipulations and attacks that are described in Section 2.2; the following paragraphs explain how S-RIP deals with these threats. However, S-RIP assumes that malicious nodes do not collude and that manipulations are only performed by single nodes.

Measures against router impersonation: S-RIP assumes that every router in a routing domain shares a secret key with every other router in the same routing domain. If an authentication algorithm is used, for example keyed MD5 (see Section 2.4 for details), a routing message can be validated by its message authentication code (MAC), which works as follows: The sender calculates the MAC from the key it shares with the other router and the message it wants to send, using the authentication algorithm. It sends the MAC along with the original message to the receiver, who then uses the shared key to calculate the MAC from the message it received. If the calculated MAC and the one it got along with the message match, the message is validated and the receiver can conclude that only the router it shares the key with can be the sender. This is because an unauthorized node does not know the key and thus would not be able to compute a MAC that the receiver can validate. This procedure resembles the concept of digital signatures (see Section 2.3), but a major difference is that MACs are computed and validated using only one key, whereas digital signatures are asymmetric cryptosystems that use a public and a private key. The fact that only one key is used makes MAC authentication more vulnerable. If an attacker knows the key or wiretaps sender and receiver while they negotiate the key, it can still impersonate a legitimate node. Furthermore, the utilization of keys makes the whole process more complex.

Measures against prefix impersonation: Without further measures a router is not able to detect false updates it receives from a neighbor who pretends to be directly connected to a subnet which in reality is either farther away or does not even exist. In S-RIP it is presumed that there is a central authority that has a global view of the autonomous system's topology and thus knows which router is connected to which subnets. This is called router prefix mapping. In an autonomous system the topology can be pre-configured on each router so that each router gains this perfect knowledge. Once a router has learned the topology, changes like additions or deletions of subnets can be distributed through a secure channel.

Measures against shorter and longer distance fraud: In distance vector routing protocols it is very difficult to verify routing updates. Therefore, S-RIP tries to approach correctness by checking if new routes are consistent with the information that is stored in the nodes from which the route was inferred. In case the information in those nodes does not contradict the new route, the route is considered valid. An example for a consistency check is illustrated in Figure 3.11. Router *B* sends a routing update to router *A*, containing a route to router

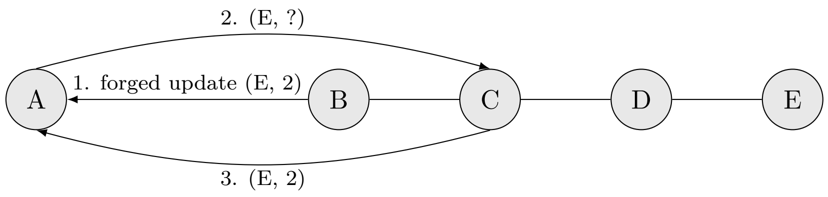


Figure 3.11: Consistency check in S-RIP

E. It claims to be able to reach *E* in only two hops, using router *C* as the next hop. In order to check the route's consistency router *A* consults router *C* and learns that *C* needs two hops to reach *E*. This is contradictory to *B*'s statement, as *B*'s route to *E* should be one hop longer than *C*'s route to *E*. *A* can conclude that either *B* or *C* is sending false information, but it cannot tell who of them lies. Thus it takes precautionary measures and discards the routing update it received from *B*. If *B* had propagated a route of three hops to *E*, this route would have been consistent with the information *C* provides and the route would not have been dropped.

In contrast to link state routing protocols, where routers know the complete network topology, in distance vector routing protocols nodes communicate only with their direct neighbors and do not store the whole topology. S-RIP strikes a balance between these two approaches in order to make consistency checks easier. It assumes that nodes reveal the next hops of all their routing table entries voluntarily to their directly connected neighbors but also to remote nodes on demand (via RIP route requests or SNMP MIB queries [WKV04]). In Figure 3.11 router *C* would tell router *A* that *D* is its next hop on a route to *E*, if router *A* would ask for that information.

It is, of course, possible that in the course of a consistency check a router wants to get routing information from a remote node but cannot reach that node because it does not know a valid route. S-RIP solves this problem by operating temporary routing tables which contain all the routes that have not yet been verified. Note that temporary routing tables are not used for routing data traffic; this is only possible after a route has been validated. Assuming that router *A* learns a route to *C* from router *B*, it first installs this route in its temporary routing table (as it cannot yet tell if the route is valid). Next, it sends a routing request destined for *C* to router *B*; *B* should know how to forward this request to *C*, as it advertised a corresponding route to *A*. As soon as router *C* receives the request it answers *A* using either its regular routing table (if it already contains a validated route for *A*) or its temporary routing table.

In "regular" RIPv2 it is not mandatory to fill in the next hop field; if it is empty, the receiver of the update uses the sender as the next hop. In S-RIP, every node uses the next hop field provided by RIPv2 messages to include its direct next hop. By this means the recipient can decide if it wants to use the sender as the next hop or if it wants to use the IP address advertised in the next hop field. This is considered an advantage as it is easier for the receiver to check if it can reach the next hop IP address directly. If the recipient cannot reach the next hop directly, it just ignores the next hop field. Moreover, finding nodes for consistency checks is easier if next hop fields are always filled in.

Reputation-Based Validation Framework: When a node advertises a route to another node it is likely that a subset of nodes already knows that piece of topology information. S-RIP takes advantage of this fact insofar that the receiving node consults these nodes in order to ensure that the route is correct. Inconsistencies can be detected by neighboring nodes which precludes wrong routes from being propagated on and on. In the reputation-based framework each node is assigned a *reputation value* by every node in the same network. It is obviously too restrictive to completely mistrust a node and too careless to entirely believe without further checking in the information a node propagates. An equation for computing another node's reputation value is established as follows:

$r_i(j, t_m)$ stands for node i's rating of node j's reputation at time t_m. It describes the reliability of j's routing information, destined for i, measured by its past behavior. In the beginning, every node assigns a starting reputation value to every other node in the same network. $c_i(j, t)$ is used by i to express the correctness of node j's routing updates at time t; in [WKV03] the authors suggest the following values:

$$c_i(j,t) = \begin{cases} 0 & \text{if } j \text{ provides incorrect information at time } t \\ 0.25 & \text{if } j \text{ provides conflicting information at time } t \\ 0.5 & \text{if } j \text{ provides consistent information at time } t \end{cases}$$

In order to give more importance to recent behavior compared to behavior further back in time, a time weighting factor $w(t)$ is introduced:

$$w(t) = \frac{1}{2^{t_m - t}} \quad 1 \leq t \leq t_m.$$

Altogether we get

$$r_i(j, t_m) = \sum_{t=1}^{t_m} c_i(j,t) \cdot w(t).$$

Using this equation it is possible to estimate the confidence that node j will send correct information to node i one step ahead in time:

$$r_i(j, t+1) = \frac{r_i(j,t)}{2} + c_i(j, t+1).$$

During a consistency check, a node will probably not consult only one other node, but several nodes. This leads to the development of a value for *accumulated confidence*. If x is a node that wants to validate a routing update, and $v_1, v_2, ..., v_n$ are nodes which node x consults during a consistency check, and $r_x(v_i)$ denotes how x rates the reputation of node v_i, then, provided that $v_1, ..., v_n$ deliver consistent information, $r_x(v_1, v_2, ..., v_n) = r_x(v[1..n])$ denotes node x's accumulated confidence in that information. It is defined as follows:

$$r_x(v[1..n]) = \begin{cases} r_x(v_1) & \text{if } n = 1 \\ r_x(v_1) + (1 - r_x(v_1)) \cdot r_x(v_2) & \text{if } n = 2 \\ r_x(v[1..n-1]) + (1 - r_x(v[1..n-1])) \cdot r_x(v_n) & \text{if } n > 2 \end{cases}$$

The more nodes with a reputation value $\neq 0$ validate a particular route, the more the route is likely to be correct. If a node with a reputation value of 0 confirms a route, this does not change the accumulated confidence. If – in contrast to that – a node with a reputation value of 1 attests the validity of a route, the accumulated confidence switches to 1 at once.

Now that the concept of node reputation has been presented, the next step is to determine how different reputation values affect the number of nodes involved in a consistency check. For this purpose two thresholds, θ_1 and θ_2, are introduced. Depending on how a node i, which received a routing update from node j, classifies j's reputation value, i takes further actions:

- **Rule 1 (Low reputation):** If node i's rating of node j's reputation is low, that means $0 \leq r_i(j) < \theta_1$, i does not cross-check the consistency of a routing update received from j with other nodes and just ignores the update.

- **Rule 2 (Medium reputation):** If node i's rating of node j's reputation is in the medium range, that means $\theta_1 \leq r_i(j) < \theta_2$, i cross-checks the consistency of a routing update received from j with other node(s). The number of nodes that are consulted depends on a *sized window* (see below).

- **Rule 3 (High reputation):** If node i's rating of node j's reputation is high, that means $\theta_2 \leq r_i(j) \leq 1$, i accepts a routing update from j without cross-checking it.

Rule 1 can help to prevent denial of service attacks. An untrusted node's routing updates are not cross-checked. As a result, a malicious node with low reputation is not able to incapacitate another node by sending out numerous fake routing updates, which the other node would normally validate. Nodes with medium reputation values can quickly go up to high reputation if they behave well and can quickly be demoted to low reputation if they misbehave. This is why initial values should be placed in the medium range. Routing updates from nodes with high reputation are not cross-checked, which involves the risk that a trusted node propagating wrong information remains undetected. There are two solutions to that problem. First, a node's reputation value could be reset to medium reputation after a specified period. That means no node will be trusted forever. Second, threshold θ_2 could be increased, resulting in the increase of time until a node reaches high reputation.

A sized window determines how many nodes are consulted in a consistency check. If all nodes that know a certain route are involved in a consistency check, the network overhead will be excessively high, but the more nodes agree with a certain route the higher the confidence that this route is correct. At the beginning of a consistency check the only node in the sized window is the advertising node of a route, that means the window size is 1. If the accumulated confidence for that route is less than threshold θ_2, the window size grows and another node is involved in the check. The increase of the window size continues until either the consistency check for the route fails or is successful. It fails if an inconsistency occurs. It succeeds if all the nodes in the sized window confirm the route and threshold θ_2 is exceeded or if all the nodes in the sized window confirm the route and all nodes that know that route have been consulted.

By now, the basic principles that are required for understanding how S-RIP actually works have been explained. In the following, we assume a simple network as depicted in Figure 3.12, where v_0 is the receiver of a routing advertisement, v_1 the advertiser of the update and v_n the destination. $dist(v_1, v_n)$ corresponds to the number of hops on a route from v_1 to v_n and $nh(v_1, v_n)$ stands for the next hop on the route from v_1 to v_n. S-RIP is only triggered if a new route results in

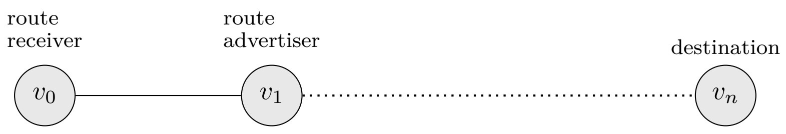


Figure 3.12: An example network in S-RIP

a route change or topology change and passes the validation process of RIPv2 and thus will be installed in the receiver's routing table. Then, S-RIP performs four checks:

Check 1 (Self-consistency check):

- The node that receives the routing update checks if it shares a secret key with the destination node. If so, the destination node is legitimate. If not, the route is rejected.
- If v_1 sends $dist(v_1, v_n) = 0$ to v_0, this indicates that the route is either for v_1 or a subnet directly attached to v_1. So the next hop on the route from v_1 to v_n should be v_1.
- If $1 \leq dist(v_1, v_n) \leq 15$, the next hop must not be v_0 or v_1.
- The next hop on a route should always be validated in order to prevent count-to-infinity (see Section 2.1.1).

Check 2 (Router/prefix authentication):

- If v_1 sends v_0 the distance $dist(v_1, v_n) = 0$, it wants to inform v_0 about a route either to itself or to a subnet which it is directly connected to.
- v_0 has two possibilities to validate that distance:
  - With the help of router-prefix-mapping v_0 can check if v_1 is directly connected to the subnet. Or, if the route was a route to v_1 itself, message authentication helps to authenticate data origin.

  - v_1 has to prove that it knows the secret key which is shared by v_0 and v_n. v_0 requests the key by sending a random number to v_1. If v_1 knows the secret key k_{v0,v_n}, it proves this by using a message digest algorithm (e.g. MD5) to compute the message digest of the random number, v_0 and k_{v0,v_n} and by sending the digest to v_0.

- If the validation is successful, v_0 accepts the route it has learned from v_1. If validation fails, it drops the route.

Check 3 (Consistency check):

- If $1 \leq dist(v_1, v_n) \leq 15$, this means that v_n is reachable from v_1. Assuming $nh(v_1, v_n) = v_2$, v_0 will ask v_2 to send back next hops and distances from v_2 to v_n and from v_2 to v_1. v_0 considers the route from v_1 to v_n consistent with the information it gets from v_2 if $dist(v_2, v_1) = 1$ and $dist(v_1, v_n) = dist(v_2, v_n) + 1$.

- v_0 computes an accumulated confidence $r_{v_0}(v_1, v_2)$ if v_1 is consistent with v_2 and checks if the result is $\geq \theta_2$. If so, it accepts the route and installs it in its routing table.

- If, however, $r_{v_0}(v_1, v_2) < \theta_2$, v_0 includes further nodes in the consistency check.

- Using the next hop information, v_0 chooses a node v_i and checks if this node has been consulted before. If so, a network loop is detected and the advertised route is dropped. If no loop is detected, v_0 sends a route request to node v_i.

- New nodes are consulted until either:
  - $r_{v_0}(v[1..k]) \geq \theta_2$. Then the advertised route is considered correct.
  - $r_{v_0}(v[1..k-1]) < \theta_2$ and $dist(v_{k-1}, v_n) \neq dist(v_k, v_n) + dist(v_k, v_{k-1})$ which means that the information from v_k is inconsistent with the information from v_{k-1}. In this case v_0 considers the route it got from v_1 inconsistent.

  – All the nodes on the route to v_n have been consulted, as well as v_n itself. If the information from v_n and v_{n-1} is conflicting, the advertised route is considered inconsistent and is rejected. If v_n and v_{n-1} do not disagree, v_0 performs router/prefix authentication with v_n, according to the description above (check 2). In case authentication is successful, the route is considered correct, regardless of the value of $r_{v_0}(v[1..n])$.

Check 4 (Infinity route):

- If v_1 tells v_0 that $dist(v_1, v_n) \geq 15$, this means that v_0 cannot reach v_n via v_1, as the maximum hop-count in RIP is 15, and a hop-count of 16 means unreachability.

- v_0 accepts the route without validating it and will not send packets destined for v_n via v_1. If the network is redundancy-free, v_0 does not know an alternative route to reach v_n.

S-RIP's operating principles can be summarized and described in pseudocode:

Algorithm 3.2: S-RIP, adapted from [WKV03], page 19

```
INPUT: $v_0, v_1, [v_n, dist(v_1, v_n), nh(v_1, v_n)], \theta_1, \theta_2$
OUTPUT: accept or reject $[v_n, dist(v_1, v_n), nh(v_1, v_n)]$

if $r_{v_0}(v_1) > \theta_2$ or $dist(v_1, v_n) \geq 15$ then
    accept $[v_n, dist(v_1, v_n), nh(v_1, v_n)]$; goto: END
end if
router/prefix authentication:
if $dist(v_1, v_n) = 0$ then
    perform router/prefix authentication of $v_n$
    if $v_1$ demonstrates the knowledge of key $k_{v_0,v_n}$ or $v_1$ is directly connected to the
            subnet then
        accept $[v_n, dist(v_1, v_n), nh(v_1, v_n)]$
    else
        reject $[v_n, dist(v_1, v_n), nh(v_1, v_n)]$
    end if
    goto: END
```

```
end if
$i = 1$
while TRUE do
    $i = i + 1; v_i = nh(v_{i-1}, v_n)$;
    request from $v_i$ : $[v_n, *, *]$ and $[v_{i-1}, *, *]$
    wait until receiving $[v_n, dist(v_i, v_n), nh(v_i, v_n)]$ and $[v_{i-1}, dist(v_i, v_{i-1}), nh(v_i, v_{i-1})]$
    if $nh(v_i, v_n) \in \{v_j\}(1 \leq j \leq i-1)$ then
        reject $[v_n, dist(v_1, v_n), nh(v_1, v_n)]$; goto: END
    end if
    if $dist(v_{i-1}, v_n) = dist(v_i, v_n) + dist(v_i, v_{i-1})$ then
        calculate $r_{v_0}(v[1..i])$
        if $r_{v_0}(v[1..i]) > \theta_2$ then
            accept $[v_n, dist(v_1, v_n), nh(v_1, v_n)]$; goto: END
        else if $dist(v_i, v_n) = 0$ then
            goto: router/prefix authentication
        end if
    else
        reject $[v_n, dist(v_1, v_n), nh(v_1, v_n)]$; goto: END
    end if
end while
END
```

The beginning of this section gives a basic overview of threats to distance vector routing protocols and of solutions to these threats. We now examine in more detail how a malevolent node or several nodes in collusion can compromise routing and how S-RIP copes with these attacks. The example network shown in figure 3.12 is still taken as a starting point.

To initiate an attack, a misbehaving router can advertise incorrect routing information. This could be wrong data about destinations, distances or next hops. S-RIP requires that every node shares a secret key with every other node in the network. Thus, a node advertising a route to a node that is not in the network or does not exist can be detected, as the node that receives the update does not share a secret key with the destination node. Wrong advertised distances can be detected in S-RIP, using router/prefix authentication and consistency checks. The first helps to reveal distances that are propagated as zero but in fact span one or more hops. The latter covers other distance fraud. A consistency check might lead to false acceptance of a route if the consulted nodes – which agree upon that route – are misled themselves by a malicious node or if some or all of them collude. However, the more nodes are consulted during a consistency check, the more the risk of falsely accepting a route decreases. If a node propagates false distances, it might also send out forged information on next hops in order to reinforce these false distances. A node v_1 that is up to that kind of fraud has two alternatives: It can either make a node up that does not even exist or claim that a node, which is in fact further away, is directly connected. In the first case, it intercepts the validation requests from node v_0 and sends back manipulated responses, claiming to be one of the nodes the requests were meant for. Nevertheless, this is not a major problem as a fictional node does not share a secret key with a legitimate node and thus can be uncovered easily. In the second case, v_1 pretends that a node that actually belongs to the same network but is not directly connected is the next hop on a route. A possible scenario is the following: v_1 needs 5 hops to reach v_n and learns that $dist(v_m, v_n) = 1$. It could then pretend to know a two hop route to v_n with v_m as the next hop. If v_m is not malicious and does not collaborate with v_1, v_0 will discover the misinformation when performing consistency checks. If, however, v_m and v_1 collude, this can be seen as a virtual connection or a "wormhole" between the two nodes, which S-RIP can only detect if it knows how nodes in a network are physically connected among each other.

Instead of propagating incorrect distances or next hops in a routing update, a malicious node can also give false information when being questioned during a consistency check. This might result in nodes rejecting actually correct new routes because the false information caused an inconsistency. Yet, this is not considered a drawback because disregarding such a route – even though it is correct – also means disregarding a route with a malicious node along the way, which could lead to problems in the future. Consequently, taking a different but maybe longer route can be beneficial in the long run. A node giving false information in a consistency check can also decline an honest node's reputation. If node v_0's reputation value for node v_1 drops below θ_1 (and thus is very low), v_0 will ignore updates coming from v_1. By advertising correct routes for some time, v_1 can regain v_0's confidence. A malicious node v_m cannot damage another node's reputation at all times. This is only possible if S-RIP is triggered, which only happens upon route changes, and if both nodes, v_1 and v_m, take part in the consistency check.

Another easy way for an adversary to interfere in a consistency check is to ignore validation requests or to not forward validation requests from v_0 to other nodes or responses from other nodes to v_0. v_0 will reject a route if it cannot validate it because it does not get any answers. If there is redundancy in the network, this problem is covered because alternate routes will be found which, in addition, might be free of fraudulent nodes.

Malicious nodes can manipulate validation messages that they are supposed to forward during a consistency check. This can only cause problems if one of the involved routers does not use message authentication. If, for example, an adversary captures and retains a validation request destined for a non-secured router, it can create a response that supports its mischievous intentions and send it back on behalf of the original receiver. The authors suggest to use Internet Protocol Security, if available, to avoid this kind of attack. Another problem can arise if a remote non-secured router answers to a validation request, and a malicious router intercepts the answer along the way and manipulates it in order to make the route look consistent although it is false. However, in case all routers use S-RIP with MD5 authentication a manipulated message will be detected.

3.6 Secure Distance Vector Routing Protocol S-DV

Although S-RIP (see Section 3.5) enhances security of distance vector routing protocols it adds a considerable amount of overhead to the routing procedure. S-DV (secure distance vector routing protocol), a new approach to secure distance vector routing protocols, developed by Babakhouya, Challal, Bouabdallah and Gharout, promises to provide a similar security level while producing less overhead [BCBG06]. The main improvement compared to S-RIP is the introduction of trusted routers (S-DV routers) that help to reveal attacks and manipulations during the routing process. A network running S-DV consists of "regular" routers and a smaller number of trusted S-DV routers. Another important difference is that the detection of manipulated routes in S-RIP is non-deterministic, whereas it is deterministic in S-DV. If routing updates are digitally signed, this helps against illegitimate routers that manipulate, fabricate or delete routing messages, but it does not help against legitimate routers or subverted routers that propagate wrong distances or unreachable destinations. To defeat the latter, it is necessary for a router to know more than only its next hop neighbors; consistency checks, as already used in S-RIP provide the required knowledge. Just as S-RIP, S-DV does not give a guarantee that the route, which is chosen in the end, is optimal. It, however, does ensure that the route is secure. For this purpose a *security indicator*, a new criterion for judging a route's secureness, is brought into use. S-DV addresses router impersonation and prefix impersonation as well as shorter and longer distance fraud and thus has the same security objectives as S-RIP (see Sections 2.2 and 3.5).

In short, S-DV uses trusted routers which offer a distance request/distance reply mechanism for checking the consistency of a route. If a trusted router detects forged routing updates it receives from a neighbor, these updates are discarded. Moreover, S-DV assigns a value representing the security indicator to each route that passes through one of its neighbors. S-DV employs symmetric key cryptography for authenticating the communication between routers. Later on, these concepts are explained in more detail. S-DV takes several assumptions:

1. Neighboring nodes share a secret key which is different for each pair of nodes.

2. S-DV nodes share a different secret key with every other S-DV node in the network. Keys are used for authentication and are distributed by any kind of key establishment mechanism.

3. Both S-DV nodes and regular nodes are aware of the subnets that are connected to their neighboring nodes. This router prefix mapping can be distributed to each router by a central authority that has the global view of the network (cf. Section 3.5).

4. A new field is added to update messages. It designates the last S-DV router that advertises or forwards a certain route and is called the *predecessor* of that route. The predecessor is supposed to make consistency checks less

complex. It is not to be confused with the predecessor used by Smith in Section 3.2.

The first three of the above assumptions are less confining compared to the assumptions taken by S-RIP: In S-RIP, every router shares a different secret key with every other router in the network and every router knows which subnets are connected to every other router in the network. This obviously saves a lot of overhead in S-DV; this subject will also be discussed later.

With the help of these assumptions the vulnerabilities in distance vector protocols can be addressed. In the following, v_j denotes the advertiser of a routing update and v_i stands for the receiver of the update. In a routing update, $dist(v_i, v_j)$ corresponds to the number of hops from node v_i to node v_j and $pred(v_i, v_j)$ designates the predecessor for that route.

Measures against router impersonation: A sequence number is added to each routing message in order to discover replay of routing messages. To prevent a malicious router from successfully taking over a legitimate router's identity, authentication is used when routing messages between two routers are exchanged. If v_j wants to send v_i a routing update, it applies a message digest algorithm to the message, the sequence number and the secret key which it shares with v_i. The result is a message authentication code (MAC) which it sends along with the routing update. Upon reception of the update, v_i first checks if the sequence number is a valid one and then calculates the MAC; if both MACs match, the message is successfully authenticated. If authentication fails or if the sequence number points to the replay of a routing message, the update is discarded.

Measures against prefix impersonation: In consequence of assumption number 3, every router knows the subnets which its neighbors are connected to. If v_j advertises to v_i a zero distance route to a certain subnet, v_i can easily check if it is one of the subnets v_j is directly connected to.

Measures against shorter and longer distance fraud: S-DV uses a special distance request/distance reply mechanism (DR mechanism) for checking the consistency of newly learned routes. An update router v_i receives from router

v_j contains the destination *dest*, the distance $dist(v_j, dest)$ between v_j and the destination and the predecessor $pred(v_j, dest)$ on that route. The predecessor v_k is a trusted router (S-DV router) and in course of the consistency check router v_i sends a distance request to the predecessor, asking it to state the distances $dist(v_k, dest)$ from the predecessor to the destination of the route and $dist(v_k, v_j)$ from the predecessor to the advertiser of the route. The sum of both distances must be equal to the distance v_j advertised in the update: $dist(v_j, dest) = dist(v_k, dest) + dist(v_k, v_j)$. Figure 3.13 gives an example: v_0, v_3 and v_5 are S-

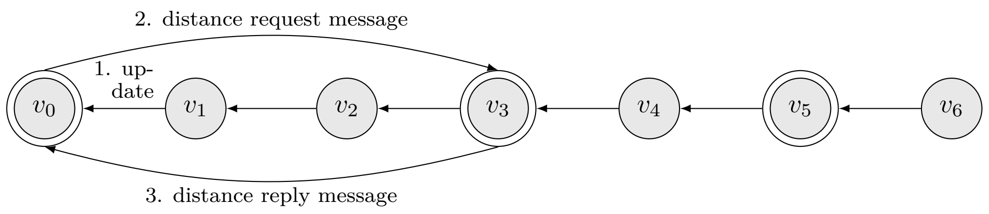


Figure 3.13: Consistency check in S-DV

DV routers, the other routers are regular RIP routers. Node v_1 sends a routing update to node v_0, containing a route for destination v_6, the predecessor of this route and the distance. It claims to be able to reach v_6 in 5 hops. If v_0 wants to validate this route it sends a distance request message to the predecessor of this route, v_3. The request contains the distances $dist(v_3, v_6)$ and $dist(v_3, v_1)$. Node v_3's distance reply message contains the requested distances $dist(v_3, v_6) = 3$ and $dist(v_3, v_1) = 2$. Node v_0 can trust node v_3 as v_3 is a S-DV node. The sum of the received distances equals the distance that node v_1 propagated in the update, and thus, v_0 concludes that the route is consistent. It then updates its routing table and propagates the route to its neighbors, increasing the hop-count by 1 and changing the predecessor from v_3 to v_0.

S-DV routers keep temporary routing tables which contain the routes that are being validated, but have not passed through the whole process yet. As soon as a route is validated it is moved to the regular routing table. If it fails the validation process, it is discarded.

Information stored in routers: For the proper functioning of S-DV, every router in the network needs to store information additionally to the routing table information. The routing table contains for each destination the identifier of that destination, the distance to that destination, the next hop on the route to the destination and the predecessor of that route. A common node (a node that is not a S-DV node) additionally stores for each neighboring node [BCBG06]:

- A secret key that it shares with this neighbor .
- A sequence number that corresponds to the last routing update it received from that neighbor.
- The subnets that are directly connected to that neighbor (router prefix mapping).

On top of this, a S-DV router v_i also maintains:

- A shared secret key for every other S-DV node in the network.
- A sequence number for the distance request/distance reply messages exchanged with every other S-DV router.
- A temporary routing table which contains all the routes that have not been verified yet and that are being checked for consistency.
- A neighbor table. It contains for every neighbor v_j:
  - v_j: The neighbor.
  - $\alpha(v_i, v_j)$: The probability that an attacker will attack the link (v_i, v_j).
  - $auth(v_i, v_j)$: The number of routing updates received from the link (v_i, v_j) that turned out to be authentic.
  - $Nauth(v_i, v_j)$: The number of routing updates received from the link (v_i, v_j) that turned out not to be authentic.
  - $coh(v_i, v_j)$: The number of routing updates received from the neighbor v_j that turned out to be consistent.
  - $Ncoh(v_i, v_j)$: The number of routing updates received from the neighbor v_j that turned out not to be consistent.

In S-DV a secure route is chosen over a short route. Thus, a suboptimal route in terms of cost is preferred if it is less likely to be attacked and if experience values convey that updates on this route were less often inconsistent or non-authentic compared to updates received on other routes. The secureness of a route is represented by the *security indicator*. A S-DV router v_i whose next hop on a certain route is its neighbor v_j computes the security indicator $Sind(v_i, v_j)$ for that route as follows:

$$Sind(v_i, v_j) = \frac{Nauth(v_i, v_j)}{Nauth(v_i, v_j) + auth(v_i, v_j)} \times \frac{Ncoh(v_i, v_j)}{Ncoh(v_i, v_j) + coh(v_i, v_j)} \times \alpha(v_i, v_j)$$

If two neighbors of v_i send v_i routes to the same destination node, v_i chooses the route whose security indicator is smaller, regardless of the route's length. Hence, it assures a higher security level for its routing operations.

S-DV routers perform several checks before they add a new route to their routing tables. Above all, a S-DV router checks if the new route is authentic. If so, it checks if the route is at all a candidate for being incorporated into its routing table. If it is, it is next checked for self-consistency. Then prefix authentication is performed, a consistency check follows and finally it is checked if the route is infinite. These checks are now explained in detail.

Authenticity check: A S-DV router v_i that receives a routing message from its neighbor v_j calculates the message's MAC and compares it to the received MAC. If they are equal, validation is successful and v_i increments the number of authentic routing updates $auth(v_i, v_j)$. If validation fails, it increments $Nauth(v_i, v_j)$ and discards the update.

Admission check: A new route is only possibly added to the receiver's routing table if at least one of the following conditions holds:

- v_i does not yet hold a route to the advertised destination in its routing table.
- The new route is shorter than the current route that v_i keeps for this particular destination.

- The new route is longer, but the route's security indicator is lower than that of the current route to the same destination.

If one of these conditions holds the following checks are performed.

Self-consistency check: Assuming v_i receives the route $[dest, dist(v_j, dest)$, $pred(v_j, dest)]$ from v_j, it examines the advertised distance. If $dist(v_j, dest) = 0$, either v_j is a S-DV node and $pred(v_j, dest) = v_j$ or v_j is not a S-DV node and $pred(v_j, dest) = null$. To avoid count-to-infinity (see Section 2.1.1) a node should not send a route back to the node it has learned that route from. Thus, $pred(v_j, dest)$ must not be v_i if $1 \leq dist(v_j, dest) \leq 15$.

Router and prefix authentication check: In case $dist(v_j, dest) = 0$ the destination of the route v_j sends to v_i is either v_j itself or a subnet that is directly connected to v_j. In the first case authentication is already assured by the authenticity check (see above). If this check had failed the route would not have reached the router and prefix authentication check as it would have been rejected before. In the latter case it can be easily verified if v_j is directly connected to the destination subnet: One assumption of S-DV is that every node in a network knows the subnets that are connected to their neighboring nodes (router prefix mapping).

Consistency check: If the destination in the route v_j advertises to v_i is reachable (that is $1 \leq dist(v_j, dest) \leq 15$), router v_i uses the distance request/distance reply mechanism, as previously explained, to verify if the route is consistent. The route is accepted without validating the consistency if there is no predecessor of that route.

Infinite route check: v_i rejects a route if $dist(v_j, dest) > 15$, because this value indicates that the destination is unreachable from v_j. There is no need to validate such a route.

After performing the above checks v_i increments $Ncoh(v_i, v_j)$ if the route failed validation. However, if it is validated, v_i increments $coh(v_i, v_j)$ and updates its routing table if the security indicator of the new route promises a higher security compared to the security indicator of the current route in the routing table.

In addition to the possible attacks that were discussed earlier, there are two kinds of attacks left to be examined. The predecessor information might be deliberately faulty or distance request/distance reply messages (DR messages) might be withheld or manipulated. By definition, predecessor nodes are always S-DV nodes. If a malicious node v_j advertises a node that is not a S-DV node as the predecessor of a route, this will be discovered by the first S-DV node that gets the manipulated routing update. According to the assumptions, every S-DV node shares a secret key with every other S-DV node in the network. If a S-DV node does not share a secret key with the pretended predecessor, this node cannot be a S-DV node and thus cannot be the predecessor.

A malicious router can also advertise a wrong predecessor, which is nevertheless a S-DV node. The first S-DV router that receives the forged update will reveal the fraud. Figure 3.14 shows an example. Node v_1 is 5 hops away from node v_6.

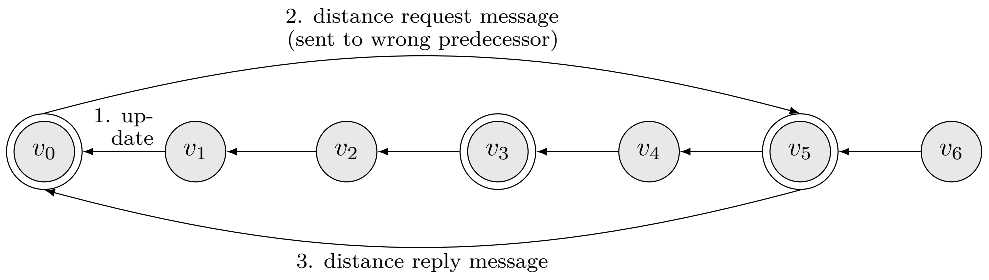

2. distance request message
(sent to wrong predecessor)
1. up-
date
v_0
v_1
v_2
v_3
v_4
v_5
v_6
3. distance reply message

Figure 3.14: Detection of predecessor fraud in S-DV

It might pretend to be v_5's neighbor and thus to be only 2 hops away from v_6. In the routing update it sends to node v_0 it might then pretend that v_5, which is a S-DV node, is the predecessor on the route to v_6, although in fact v_3 is the predecessor. When v_0 receives the update it exchanges DR messages with the claimed predecessor v_5, as v_0 itself is a S-DV router. v_0 requests the distances $dist(v_5, v_6)$ and $dist(v_5, v_1)$ from v_5. Router v_5 knows that it is 4 hops away from v_1, and not 1 hop, as v_1 pretends. Hence, it sends $dist(v_5, v_1) = 4$ to v_0 and v_0 detects the fraud as $2 \neq dist(v_5, v_6) + dist(v_5, v_1)$.

The alteration of DR messages on their way from sender to receiver can be easily detected because message authentication codes are sent along with them (also to prevent router impersonation). If the receiver gets a modified DR message and uses a message digest algorithm to compute the MAC, this MAC will not equal the received MAC. A malicious router might also choose not to forward DR messages, which results in the subversion of consistency checks. In this case, a route that is in fact correct is dropped. However, a new route will be found eventually, as routing information is exchanged on a regular basis.

A comparison between the overhead produced by S-DV and S-RIP is drawn in Chapter 4.

3.7 Symmetric Key Based Techniques

In [BKPD09] Bruhadeshwar et al. suggest the use of symmetric keys for securing routing protocols and present solutions for BGP, OSPF and RIP. Here, only the approach concerning RIP is examined. The authors distinguish between attacks in the control plane, where routers exchange routing information, and attacks in the data plane, where packets are forwarded. During the exchange of routing information, an adversary can alter routing messages that it is supposed to forward, or even feed in bogus information; both attacks result in wrong computation of optimal routes in the routing domain. The attacker can either propagate shorter distances to certain destinations in order to attract network traffic which it can spy on, or it can propagate longer distances in order to avoid traffic which results in cost reduction (see also Section 2.2 for details on shorter and longer distance fraud). In the actual data phase a malicious entity can send packets along routes different from the optimal routes that were computed in the control plane. So even if it cooperated in the first place, it can now choose routes at its will. This is why it is not sufficient to merely protect data transmission in the control plane [BKPD09]. Attacks in distance vector protocols are difficult to uncover as none of the participating nodes has the global view of the network topology. Moreover, malicious nodes can collude in order to reinforce the wrong information they propagate. This makes it even more difficult to detect attacks, even though this approach can handle single misbehaving routers as well as a set of misbehaving routers.

As a consequence of the previous statements, the authors propose a solution that combines security in the control *and* data planes. A combined solution conveniently results in reduced overhead. The approach uses a set of symmetric keys to support an authenticated query-response mechanism that helps to find inconsistencies in propagated hop-counts. Yet, it does not offer the possibility of recovering from an inconsistency once it has been detected. For RIP, distributed key distribution protocols are used, that means there is no central authority and each router or autonomous system manages the distribution of symmetric keys to other parties on its own.

To illustrate the above mentioned key distribution protocol it is best to picture a star communication network with one node in the center and a set of satellite nodes that surround the center node (see Figure 3.15). The center node can

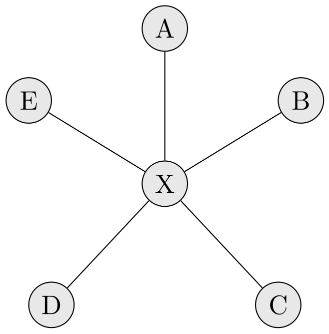


Figure 3.15: A star communication network

exchange messages with each satellite node. Unlike that, the satellite nodes can only communicate with the center node and not among each other. Depending on the number of satellite nodes, the center node creates a certain number k of symmetric keys. There must be enough keys so that the center node can share a unique subset of keys (subset size = l) with each satellite node. If the subset size is 2, in our example 4 keys are sufficient to make up 5 different subsets of keys ($\binom{4}{2} = 6$). Table 3.1 shows a possible key distribution. For example, node A shares secret keys $K1$ and $K2$ with the center node X. If the center node wants to communicate with a satellite node, it computes message authentication codes with each key and sends them along with the original message. The receiving satellite node uses the secret keys it shares with the center node to verify the corresponding message authentication codes.

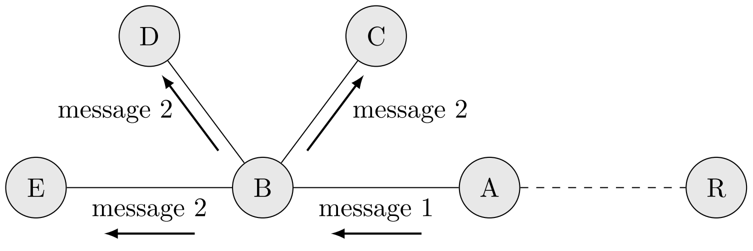
Key distribution is usually done when a router joins a network. The new router

X	K1, K2, K3, K4
A	K1, K2
B	K1, K3
C	K2, K3
D	K2, K4
E	K3, K4

Table 3.1: Key distribution

is then considered as the center node and all other nodes are considered as satellite nodes. In RIP, new updates have to be processed every 30 seconds, signature and verification in the distributed key distribution protocol are regarded as fast enough to keep pace with that. The solution presented by the authors covers shorter and longer distance frauds. To prevent both, new distances that are propagated by a router need to be confirmed by its neighbors.

Assuming that router B can reach another router R in $dist_{BR}$ hops (see Figure 3.16), it sends out a routing update $(R, dist_{BR})$ to all its directly connected neighbors. It signs the message with each of its keys separately in



message 1 $= Sign_{K_A}(R, dist_{BR} - 1)$

message 2 $= ((R, dist_{BR}, Sign_{K_B}(R, dist_{BR}))(R, dist_{BR} - 1, Sign_{K_A}(R, dist_{BR} - 1)))$

Figure 3.16: Sending of signed messages

order to prevent manipulation. Node B's set of signatures is represented by $Sign_{K_B}(R, dist_{BR})$. The next hop on the route from B to R now has to confirm the distance $dist_{BR}$ in order to validate the update. Prior to sending the update, B must have learned it from one of its neighbors, say A, which is one hop closer to R than B is (its hop length to R is thus $dist_{BR} - 1$). The message B received from A also contains A's signature for this particular message,

$Sign_{K_A}(R, dist_{BR} - 1)$. Router B attaches this signature to its message and sends $((R, dist_{BR}, Sign_{K_B}(R, dist_{BR}))(R, dist_{BR} - 1, Sign_{K_A}(R, dist_{BR} - 1)))$ to its neighbors. Another neighbor of B, for example C, can verify this update easily upon receipt. C only has to verify the signatures of B and A. When C forwards the route, it also adds its own signature and transfers B's signature. This method for authenticating updates fails if neighboring nodes collude, but it can be extended in such a way that it encloses nodes which are 3 or more hops away.

The use of symmetric keys also helps to validate the correctness of hop-counts in the data plane, where actual data is transmitted. Even if a route has been validated in the control plane, a malicious entity can decide to use a different route in the data plane which results in different hop-counts compared to the control plane. An authenticated query-response mechanism is proposed to detect this kind of misbehavior. If a router A that wants to send out data would like to verify its k-hop neighborhood, it creates a list that contains its k-hop neighbors and signs it with its keys. Then, this list and all signatures are broadcasted to every node in the network. If any node that is supposed to be in A's k-hop neighborhood receives this list and detects that it is actually not k hops away from A, it will inform A about the inconsistency. Once A has learned about an inconsistency it can correct it.

4 Simulations and Comparison of Approaches

This chapter examines the computational and storage overheads that the approaches to secure distance vector routing algorithms that are presented in Chapter 3 entail. Simulation results are summed up and the effectiveness, advantages and disadvantages of approaches are compared as far as practicable.

Implicit path method: The solution Smith proposes in his master's thesis includes the use of digital signatures and sequence numbers for routing information as well as the introduction of predecessor information (for reconstructing the route that an update implies) and destination link costs. Smith states that his approach offers protection at a cost and level that can be compared to link state security proposals, which are generally less expensive in storage space and computation time.

Predecessor information is used for path traversal which verifies the integrity of a newly learned route and the associated distance. Thus, computational overhead arises from carrying out path traversal and computing the new above mentioned fields that are added to routing updates and messages. To be specific, every time a new routing message is produced its digital signature and sequence number have to be computed and the receiver of the message must verify both. For each update the digital signature must be computed once for each link, as the predecessor of a routing entry is different for each outgoing interface. The receiver of an update has to verify the signature if the route it selects includes this particular link in its implicit path. Moreover, every time a router chooses a new route to reach a destination it has to perform path traversal.

The new fields also implicate storage overhead. Per routing message, a 64 byte digital signature and a 4 byte sequence number are added. Routing updates require even more additional storage space. 4 bytes are added for storing the sequence number, one byte for the destination link cost. Predecessor network address, predecessor network mask, IP address of the router's interface on the predecessor network and IP address on the router's interface on the destination network each require 4 bytes of storage space and 64 bytes are needed for the digital signature. This makes a total of 85 bytes of additional storage space per update.

RIP-TP: RIP with Triangle Theorem Checking and Probing uses two triangle theorem checks to detect suspicious new routing updates and probing messages to verify questionable distances (for details see Section 3.3). Compared to standard RIP, additional overhead is generated because RIP uses neither of these checks. When a router receives a new distance to a certain destination from one of its neighbors it checks if the hop count to that destination which it keeps in its own routing table is equal to or less than the number of hops from the neighbor to the destination plus 1. If this holds true, the route is accepted and one addition of two 4 byte numbers and one comparison of two 4 byte numbers constitutes the only computational overhead produced by RIP-TP. A flag bit which indicates that the distance is validated is set, this bit constitutes the storage overhead. If, however, the check does not hold true, a second check and consequently a second addition and comparison is necessary. Furthermore, this potentially or probably invalid distance is then verified by sending a probing message to the destination. Sending one probing message results in an overhead that equals sending one UDP packet and receiving one or zero ICMP messages.

In order to minimize overhead, two optimizations are introduced. First, the number of probing messages sent per update message is limited. In the worst case, 25 distances per update message would fail check 1 and/or check 2 and thus would be subject to verification by sending probing messages. Factor C reduces the number of probing messages per received update message, depending on the desired security level. The second optimization is the possibility to use a 2 byte reserved field in an update message that contains the number of update entries that were successfully verified. Using the field allows a router to share its verification results with its neighbors, who in turn do not need to check the already verified distances again. The results of failed verifications are neither stored nor propagated.

The authors run various simulation tests using the IRLSim simulator [TNWZ00]. They use different scenarios to contrast the behavior of RIP-TP with the behavior of RIP-RP, which is a variant of RIP that uses random probing. Standard RIP is left out in the comparison because it accepts and propagates even invalid distances. In RIP-RP though, a router tries to verify K routing entries which it picks randomly from each routing update. K is a configured parameter and simulations are run for $K = 1, K = 2$ and $K = 3$. In contrast to RIP-TP, RIP-RP does not share its verification results with its neighbors and it does not use any optimization mechanisms. It skips routes with infinite distances when choosing candidates for verification.

Several new variables are introduced:

- K = Number of distances per update which a router chooses randomly for verification.
- N = Node degree, i.e. number of neighboring nodes a router is connected to. $N * N$ is the number of nodes in the network.
- I = Number of distances which a malicious node randomly chooses from its routing table before decreasing their distance by 1 and sending these distances to its neighbors in a routing update.
- P = Probability with which a randomly chosen link fails every second. After a link fails, its probability to recover is 0.5 every second.
- M = Number of invalid distances (potentially invalid or probably invalid).
- L = Number of invalid distances that are revealed.
- $D = L/M$. Detection rate and measure for effectiveness.
- O = Overhead. Number of probing messages divided by number of received update messages.

In the simulation runs, there is a single malicious router in the network which is located in the center of the network. It selects I routes and manipulates the distances as described above, but it does not delete routing entries. Figures 4.1 -

4.6 show how overhead and detection rate in RIP-RP and RIP-TP change when the number of manipulated distances, the probability for link failure or the node degree changes, respectively.

Figure 4.1 shows that the detection rate in RIP-TP stays constantly above 0.95 regardless of the number of invalid entries per update message if the probability for link failure is set to 0.1 and the node degree is set to 4. In RIP-RP detection rates start between 0.64 and 0.72 (depending on the number K of destinations which are subject to verification) when the number of invalid entries per message is 1 and rise to 0.78 - 0.83 when the number of invalid entries per message is 8.

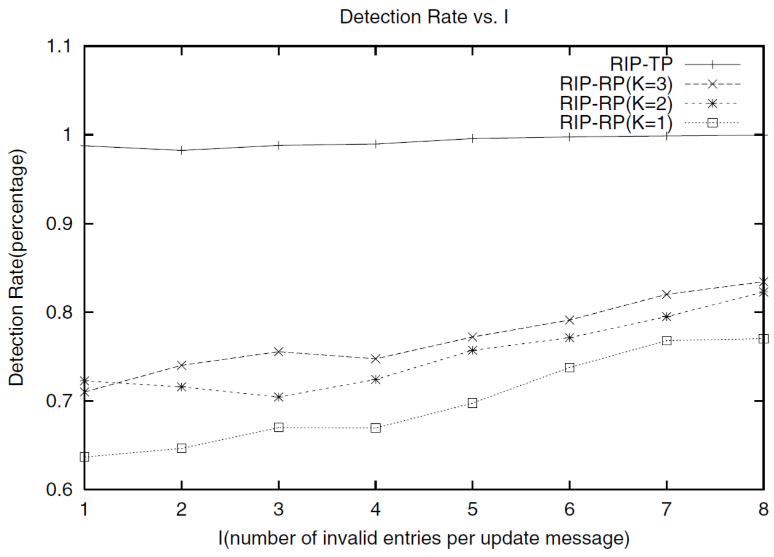


Figure 4.1: Detection rate vs. number of invalid entries per message in RIP-TP, $N = 4$, $P = 0.1$. Source: [PMZ03], Chapter 4, page 5

Figure 4.2 shows that the overhead in RIP-RP remains almost constant as the number of invalid routing entries per message increases. If the number of destinations to be verified is 1, overhead is about 0.8 and it increases by 0.4 each time the number of destinations to be verified is increased by 1. By contrast, the overhead in RIP-TP increases with an increasing number of invalid routing entries per message. If only a single routing entry is invalid, overhead is about 0.6 and it grows to about 1.0 if 8 routing entries are invalid. This is because more probing messages are triggered if the number of potentially or probably invalid distances goes up.

Figure 4.3 compares the detection rates in RIP-RP and RIP-TP for increasing

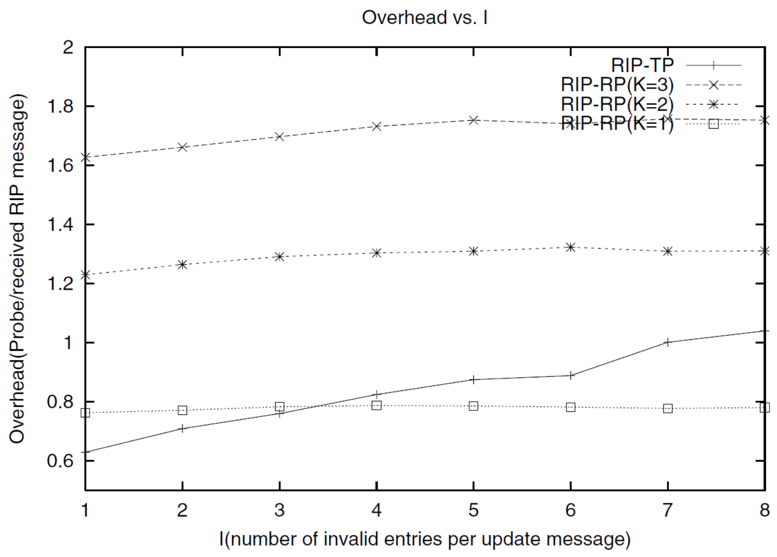


Figure 4.2: Overhead vs. number of invalid entries per message in RIP-TP, $N = 4$, $P = 0.1$. Source: [PMZ03], Chapter 4, page 5

link failure probability. If the link failure probability grows from 0.02 to 0.2 per second, RIP-TP's detection rate drops, if only slightly, from 1.0 to 0.96. This is because new routes have to be chosen if a link fails; these new routes can be potentially invalid and the triggered probing messages and ICMP reply messages might get lost. RIP-RP's detection rates increase as link failure probability increases, but they are always below 0.8, which means that RIP-TP's detection rate is always higher.

Figure 4.4 shows that a higher probability for link failures in RIP-TP does not only decrease the detection rate but also increases overhead, as more probing messages have to be sent. In RIP-RP, overhead decreases with a growing link failure probability, but is generally significantly higher than overhead in RIP-TP.

Figures 4.5 and 4.6 compare dependency of detection rates and overhead on network size. In RIP-TP, the detection rate stays almost the same if the network size grows; it is nearly 1.0 if link failure probability is set to 0.1 and the number of manipulated routing entries is set to 3. In RIP-RP detection rates vary slightly but are generally in the range between 0.6 and 0.75. RIP-TP's overhead decreases as the network size grows, whereas RIP-RP's overhead increases. With a maximum value of 0.9, RIP-TP's overhead is remarkably lower compared to

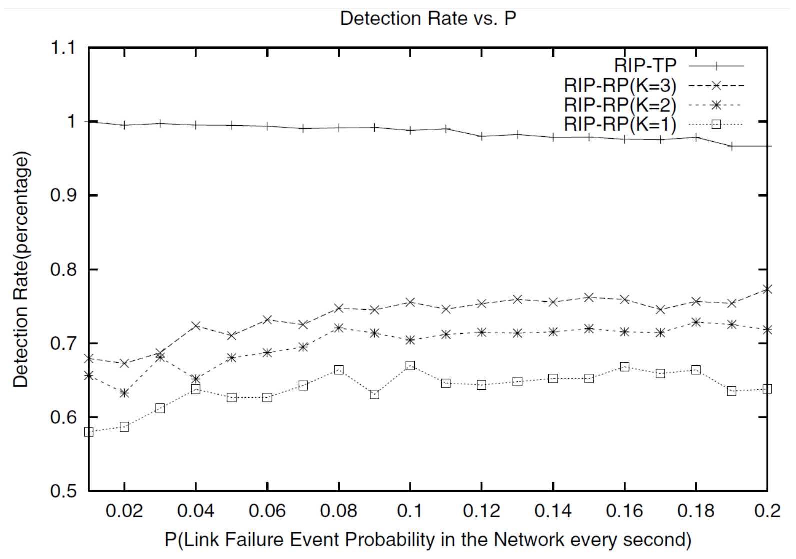


Figure 4.3: Detection rate vs. link failure probability in RIP-TP, $N = 4$, $I = 3$. Source: [PMZ03], Chapter 4, page 5

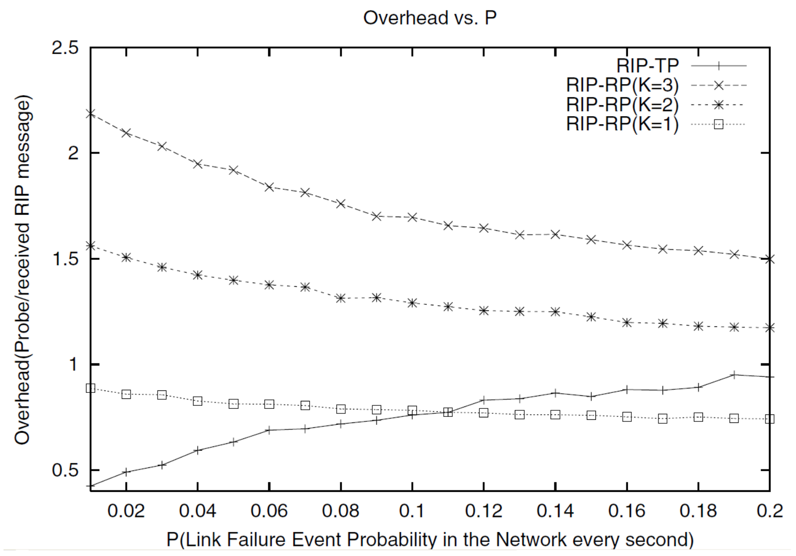


Figure 4.4: Overhead vs. link failure probability in RIP-TP, $N = 4$, $I = 3$. Source: [PMZ03], Chapter 4, page 5

RIP-RP's overhead which reaches 1.9 if the node degree is 7 and the number of invalid update entries is 3.

To sum up simulation results it can be stated that RIP-TP's detection rate never drops below 0.95 and that overhead never exceeds 1.1.

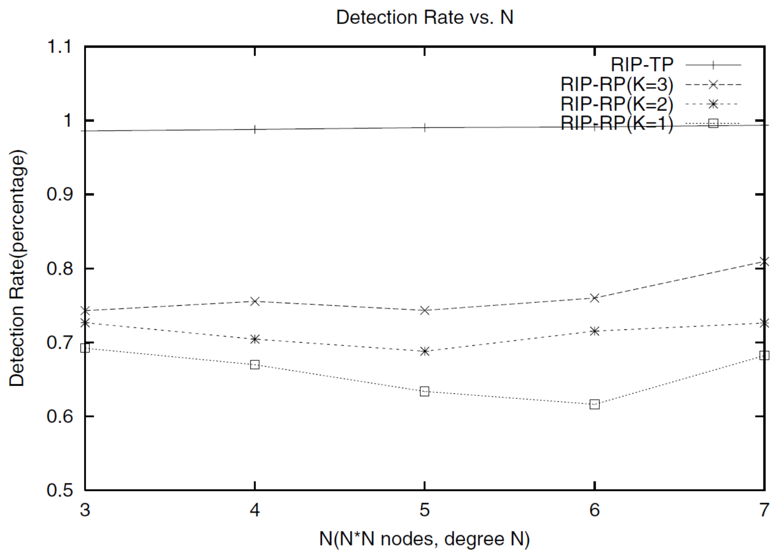


Figure 4.5: Detection rate vs. network size in RIP-TP, $P = 0.1$, $I = 3$. Source: [PMZ03], Chapter 4, page 6

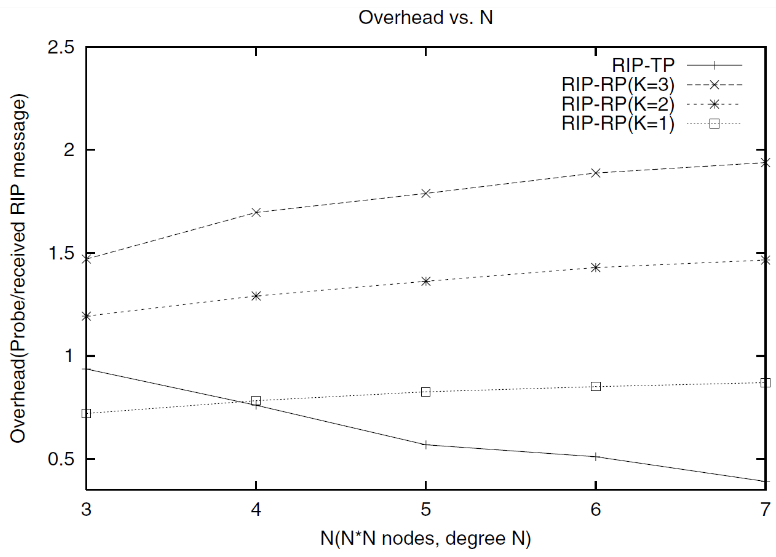


Figure 4.6: Overhead vs. network size in RIP-TP, $P = 0.1$, $I = 3$. Source: [PMZ03], Chapter 4, page 6

PAIR: PAIR can detect malicious routing updates and even recover from the damage they bring about if certain conditions are fulfilled (see Section 3.4). The possibility of recovery distinguishes PAIR from the other approaches that are discussed in Chapter 3, as it is the only one that offers such an option. PAIR is comprised of four phases, they all run in $O(n)$ time. In the first phase, the receiver of an update constructs a distance vector tree based on the predecessor information which it gets in the update. Next, it updates hop lengths, path sums and net path sums of all the nodes according to the tree. Then it checks if the net path sum of at least one of the nodes differs from 0. If so, an inconsistency is detected and if certain conditions are met, phase four, the recovery procedure, is initiated.

In [CM03], the authors conduct simulation runs using the network simulator ns-2 [ns211]. During simulation, not only the performance of PAIR is evaluated, it is also compared to the performance of consistency checks (CC) as used by Smith in [Smi97]. More precisely, PAIR is compared to the computation of implicit paths where a path is traced back from the destination to the source of the update when a node gets a routing update from one of its neighbors. CC cannot detect an inconsistency when a router on the way manipulates an update while keeping the network topology in mind. Also, performance studies for recovery are not compared to CC because CC does not offer the possibility of recovering from false updates. Here, simulations are carried out to analyze

- the probability that a forged routing update will be detected (which is 0 in RIP),
- the probability that the system can be recovered after a forged update has been received (which is 0 in RIP and CC) and
- the malicious distance, which is the average number of entries that have to be changed in an update to make the update consistent.

In the simulation setup

- network topologies are generated at random,
- the default value for the average node degree is 4, for the number of nodes it is 40 and the link cost is 1 and
- malicious updates are either created randomly or in such a way that detection probability is minimized.

Figure 4.7 shows how detection probability depends on node degree if for one entry in the update message the pair predecessor/path sum is randomly altered (in PAIR) or if the pair predecessor/hop length is altered (in CC). It is

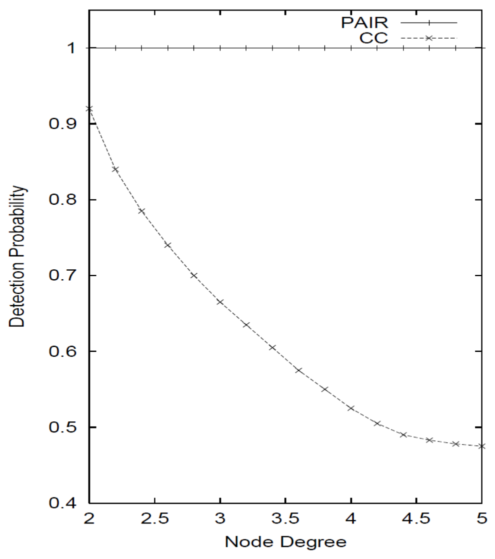


Figure 4.7: Detection probability vs. node degree in PAIR. Source: [CM03], Chapter 3, page 5

immediately obvious that PAIR's detection probability is independent of node degree and that it is almost 0.99 whereas CC's detection probability decreases with growing node degree. This is because CC will not detect a manipulated predecessor/ hop length pair if the node in question is a leaf of the distance vector tree.

If the number of randomly chosen manipulated entry pairs increases, PAIR and CC react differently, as can be seen in Figure 4.8.

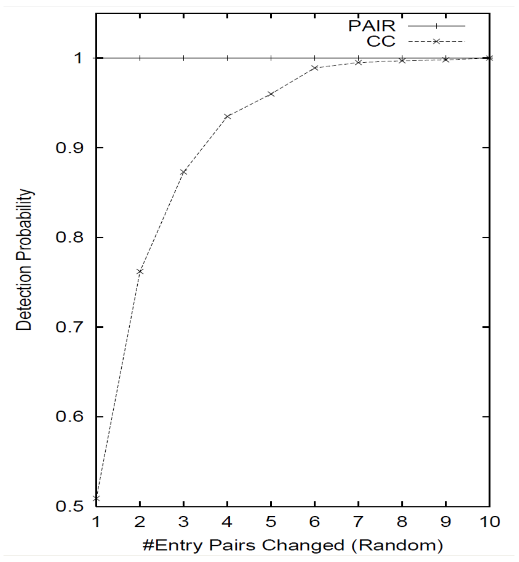


Figure 4.8: Detection probability vs. number of entry pairs changed in PAIR. Source: [CM03], Chapter 3, page 5

PAIR's detection probability does not change as the number of manipulated entry pairs in an update increases and is again almost 0.99. The detection probability of CC improves the more entry pairs are changed. If only one entry pair is changed, CC's detection probability is 0.5 but it quickly converges to PAIR's detection probability; from 8 changed entry pairs onwards their detection probability is almost equal. The behavior of CC can be explained as follows: In this setup, about half of the nodes in a distance vector tree are leaf nodes, and CC cannot detect false updates if the node in question is a leaf node. This means that the probability for CC to detect manipulations is about 50% if one entry pair is changed. The more entry pairs are changed, the higher the probability that non-leaf nodes are affected, which means that the probability for CC to detect these manipulations grows quickly.

Simulation runs that examine the dependency of distance (average number of update entries that have to be corrected in order to make the update consistent) on node degree show that PAIR's distance is generally about twice as high as CC's distance. Both distances decrease quickly with increasing node degree (see Figure 4.9). The reason for this behavior is that the higher the node degree, the denser the network and thus the higher the probability that two nodes are directly connected. Another fact that is worth noting is that for both algorithms distance grows as the number of nodes in a network increases.

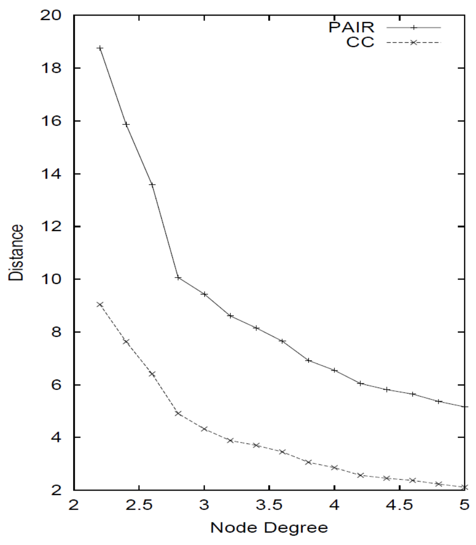


Figure 4.9: Distance vs. node degree in PAIR. Source: [CM03], Chapter 3, page 5

Figures 4.10 and 4.11 depict the dependency of recovery probability on node degree and network size. CC is not included in both simulation runs as it does not offer the possibility of recovery. In summary, it can be stated that PAIR's recovery probability never falls below 0.8.

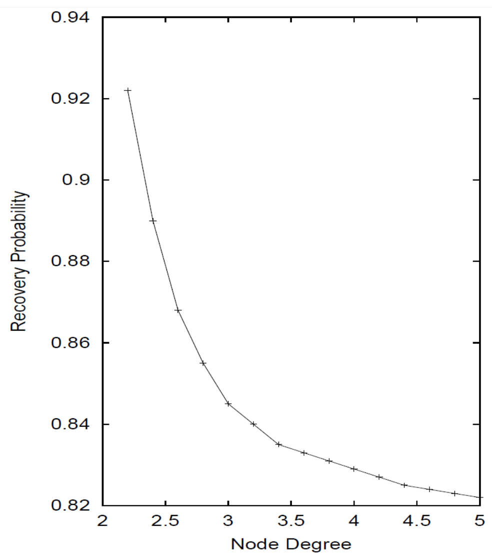


Figure 4.10: Recovery probability vs. node degree in PAIR. Source: [CM03], Chapter 3, page 5

The higher the node degree, the smaller the height of the distance vector tree which makes recovery more difficult.

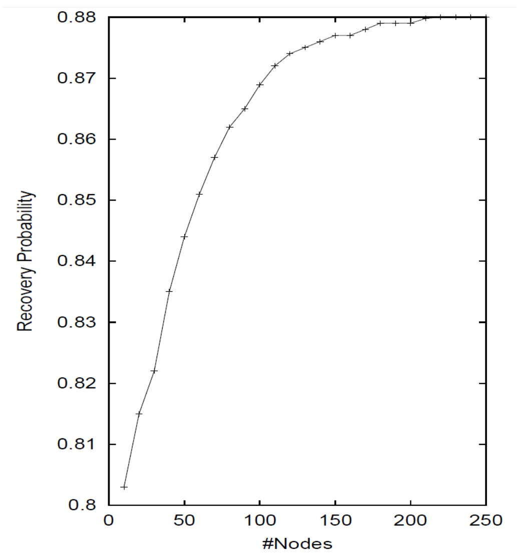


Figure 4.11: Recovery probability vs. number of nodes in PAIR. Source: [CM03], Chapter 3, page 5

The more nodes in the network, the less likely loops are created which makes recovery easier.

S-RIP: In S-RIP, a new route is considered correct if the consistency check provides a positive result. It can help to avert distance fraud and router and prefix impersonation. However, if a route is rated valid, this does not mean that it is optimal, too. This can be seen as a drawback, but on the other hand it provides a compromise between efficiency and security. If a large number of nodes are included in a consistency check and if they all agree upon the correctness of the examined route, the confidence that this route is in fact correct is high, but network overhead also grows with the number of consulted nodes. A sized window (for details see Section 3.5) helps to determine how many nodes to consult in a consistency check, dependent on which factor more importance is ascribed to – security or efficiency.

In their simulation studies and efficiency analysis, the authors distinguish between two cases: They compute routing overhead for the maximally secured case in which all other nodes are included in a consistency check, whereas they simulate cases with different threshold settings. In a scenario with n routers, m subnets and an average route length of $l + 1$ hops, routing overhead in a maximally secured case – where every router validates every route with all routers that propagated that route – is computed as follows: Assume router v_1 sends an update to router v_0 and v_0 wants to validate the route's consistency with all other nodes on that route (illustrated in Figure 4.12). It does not send a

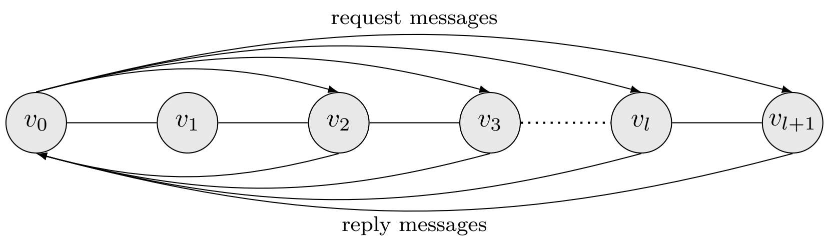


Figure 4.12: Messages sent in a maximally secured case in S-RIP

request message to v_1 because that is the node it received the update from. The first request is sent to v_2, the next to v_3 and so on, until the last hop on that route, v_{l+1}, is reached. Each of the questioned nodes sends back a reply message. Thus, the total number of messages sent is $2 \cdot l$. The message sent to node v_2 needs 2 hops to reach its destination and the answer v_2 sends back to v_0 also needs 2 hops. The second request message, destined for v_3, travels 3 hops, and so does the reply message. The last request message, destined for v_{l+1} travels $l + 1$ hops, and so does the reply message. Altogether, the number of hops (message transmissions) that all messages travel is $2 \cdot [2 + 3 + \cdots + (l + 1)] = (3 + l) \cdot l$[1]. If each router wants to validate a route for each subnet in the network, the total number of hops required for all messages is $(3 + l) \cdot l \cdot m$ for each router. As we assume that there are n routers in the network this makes a total of $(3+l)\cdot l\cdot m\cdot n$ message transmissions in the whole network.

For the additional validation overhead in bytes, this means: Each route request or response has two entries, the first one for the route from the questioned node to the destination and the second one for the questioned node to its pre-

[1]In their paper, the authors erroneously state that $2\cdot[2+3+\cdots+(l+1)] = (1+l)\cdot l$. This error is propagated in the course of their paper but it is corrected here to $2\cdot[2+3+\cdots+(l+1)] = (3 + l) \cdot l$.

security level	θ_1	θ_2
maximally secured	0	1
partially secured-1	0.1	0.9
partially secured-2	0.2	0.8
partially secured-3	0.3	0.7
not secured	0	0

Table 4.1: Different threshold settings for simulation in S-RIP

decessor. Route requests and responses are regular RIP messages with a header that consists of 25 bytes and routing entries that need 20 bytes of storage space each, which amounts to 64 bytes for a RIP message with two routing entries. Together with a 8 byte UDP header and a 20 byte IP header a packet's size is 92 bytes. As $(3 + l) \cdot l \cdot m \cdot n$ message transmissions are needed to maximize security, the total overhead is $92 \cdot (3 + l) \cdot l \cdot m \cdot n$ bytes. This is obviously an excessive amount of overhead that can only be justified in very small networks. Two optimizations are as follows: First, routing requests that are destined for the same advertiser and have the same next hop can be sent in a single message. Second, thresholds θ_1 and θ_2 can be adjusted to reduce overhead; however it is to mention that security will suffer. Simulations show how overhead changes for different values of θ_1 and θ_2.

To simulate the behavior of S-RIP depending on different values for the two thresholds and concerning security and efficiency, S-RIP was implemented in the network simulalator ns-2 [ns211]. The simulation setup is designed as follows: The network consists of 50 routers and 82 network links. In each simulation run, 5, 10, 15, 20 and 25 routers are randomly selected that commit shorter and/or longer distance fraud. More precisely, each fraudulent node randomly manipulates one of its routing entries every 2.5 seconds. Simulations are run for five different threshold settings (see Table 4.1) and take 180 seconds each. High and low node reputations are reset to medium reputation after 2 seconds.

In the first test, illustrated in Figure 4.13, the percentage of S-RIP overhead (compared to the total routing overhead) is interrelated to the percentage of misbehaving nodes in the network. The first fact that strikes attention is that in a non-secured network, there is no S-RIP overhead at all. This is reasonable because in a non-secured network S-RIP is never triggered. As already discussed

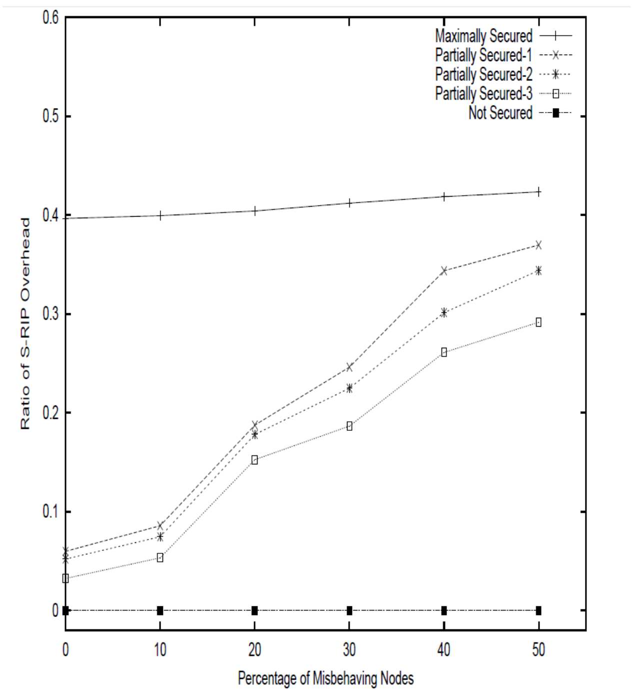


Figure 4.13: Ratio of S-RIP overhead vs. percentage of misbehaving nodes. Source: [WKV03], Chapter 4, page 12

above, one can see that the ratio of S-RIP overhead is very high – about 40% – when the network is maximally secured. The percentage increases only very slightly as the number of misbehaving nodes increases, because in a consistency check, every node has to send a request message to every other node on a route. The three partially secured scenarios react similarly to an increase in the number of misbehaving nodes. S-RIP overhead is less than 9% in all partially secured scenarios when 10% of the nodes in the network are misbehaving. In contrast to the maximally secured case S-RIP overhead here increases considerably with a growing percentage of misbehaving nodes. This is because in contrast to the maximally secured network, a lot less nodes participate in a consistency check, which means that consistency checks finish faster. As a result, more consistency checks are carried out if there are more malicious nodes, which entails more S-RIP overhead. The closer thresholds θ_1 and θ_2 lie together, the smaller the range for nodes with a medium reputation. This means that more routes will be accepted without further checking and more routes will be dropped without cross-checking; both results in less S-RIP overhead.

Figure 4.14 shows the correlation between the fraction of routes accepted for trust and the percentage of misbehaving nodes. As it is intuitively comprehensi-

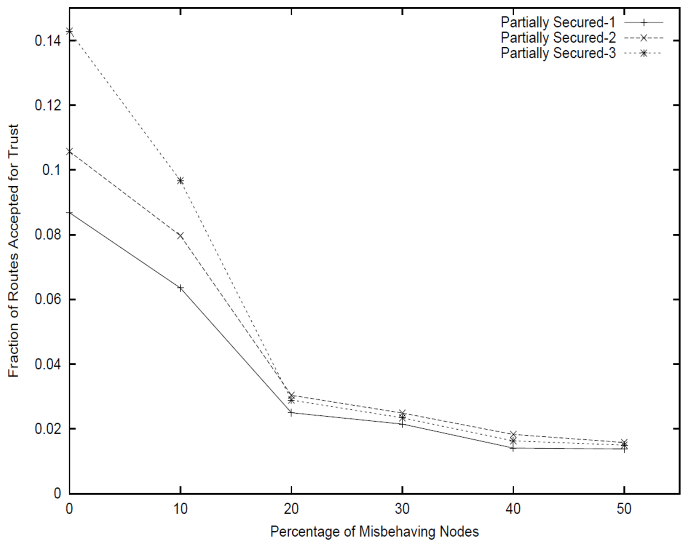


Figure 4.14: Fraction of routes accepted for trust vs. percentage of misbehaving nodes in S-RIP. Source: [WKV03], Chapter 4, page 12

ble, the risk of accepting a malicious route increases when threshold θ_2 decreases. The reason behind this is that nodes reach a high reputation more quickly and fewer consistency checks are performed. One can also see that the fraction of routes accepted for trust increases significantly more when the percentage of misbehaving nodes drops below 20%. This can be explained in the following way: The less malicious nodes per network, the more nodes will have a high reputation value and thus they will be trusted. The risk of accepting a malicious route drops below 3% in all three partially secured scenarios as soon as the percentage of malicious nodes exceeds 20%. However, the risk of rejecting correct routing updates increases. Maximally secured networks and non-secured networks are not included in this simulation as in the first case no routes at all will be accepted for trust and in the latter case all routes will be accepted without further checking.

S-DV: While S-DV offers the same benefits as S-RIP, namely protection against router and prefix impersonation as well as protection against shorter and longer distance fraud, Babakhouya et al. demonstrate in [BCBG06] that S-DV produces significantly less overhead than S-RIP. To begin with, the assumptions S-DV takes are less strong than those of S-RIP. In S-RIP, every single node in a network shares a secret key with every other node. In S-DV however, routers share secret keys only with their neighbors and only S-DV routers share a secret key with every other S-DV router in the same network. This improvement makes key management in S-DV easier compared to S-RIP. Router prefix mapping is also less strong than in S-RIP: In S-DV, a node only needs to know which subnets are attached to its neighboring routers, but in S-RIP, a node needs to know which subnets every other node of the network is connected to.

As already shown above, the number of messages in a consistency check produced by S-RIP in the maximally secured case is very high and not efficient. S-DV offers an improvement insofar as request and reply messages are only sent to respectively from S-DV routers, which reduces the amount of messages considerably. A major difference between S-RIP and S-DV is that the detection of malicious updates in S-DV is deterministic, whereas it is non-deterministic in S-RIP if node reputation is considered (that being the case in all networks that are not maximally secured). This is why S-DV is compared to S-RIP in a maximally secured network here. Three new parameters are introduced to make comparison easier:

- $\Phi(dest)$: Average number of generated message transmission by a node in a consistency check, i.e. number of hops.
- $\varphi(dest)$: Number of nodes involved in a consistency check.
- $\Psi(dest)$: Average number of generated message transmissions by all nodes in a consistency check, i.e. $\Phi(dest) \cdot \varphi(dest)$.

As stated above, the number of messages sent during a consistency check in S-RIP is $2 \cdot l$ if the average length of a route is $l + 1$. Moreover, the number of transmissions in S-RIP is $2 \cdot [2 + 3 + \cdots + (l + 1)] = l \cdot (l + 3)$. Thus, $\Phi_{SRIP}(dest) = l \cdot (l + 3)$.

In S-DV, a S-DV node sends a request message to a route's predecessor in a consistency check. Assuming that two S-DV nodes are on average k hops apart on a route, the number of transmissions is $\Phi_{SDV}(dest) = 2 \cdot k$. To make comparison between S-RIP and S-DV easier it is assumed that $k = l$, although l is usually bigger than k. Thus, $\Phi_{SDV}(dest) = 2 \cdot l$.

In S-RIP, all nodes in the network participate in a consistency check, whereas only S-DV nodes are affected by consistency checks in S-DV. If a network consists of n nodes, S of them being S-DV nodes, $\varphi_{SRIP}(dest) = n - 1$ and $\varphi_{SDV}(dest) = S - 1$. The first node on a route is not included in a consistency check in each case.

The total overhead entailed by a consistency check is represented by Ψ, which is the product of Φ and φ. Thus, $\Psi_{SRIP}(dest) = l \cdot (l + 3) \cdot (n - 1)$ and $\Psi_{SDV}(dest) = 2 \cdot l \cdot (S - 1)$.

Altogether, if S-DV instead of S-RIP is implemented, a great amount of overhead can be saved. To be more specific that is:

- $(l \cdot (l + 3)) - (2 \cdot l) = l \cdot (l + 1)$ less message transmissions initiated by a node during a consistency check.
- $(n - 1) - (S - 1) = n - S$ less nodes included in a consistency check.
- $l \cdot (l + 3) \cdot (n - 1) - 2 \cdot l \cdot (S - 1)$ less message transmissions in the whole network during a consistency check.

Due to the reduced overhead, S-DV can very well be deployed in large scale networks. At first sight it might seem like a drawback that S-DV does not guarantee that the preferred route to a destination is the shortest one. But by introducing the security indicator it chooses security over short distance.

Symmetric key based techniques: Bruhadeshwar et al. make use of symmetric keys to secure routing and state that it is important not only to secure data in the control plane where routes are computed, but that it is also important to secure data in the data plane where the transmission of the actual data takes place. The reason for that is that a malicious router might act unsuspiciously when routes are computed but it might not use these routes when actual data is forwarded. This solution offers prevention from and detection of attacks in both planes but it does not offer the possibility of correcting negative effects after a deceiver was successful.

Using a distributed key distribution protocol, every router is assigned a set of secret keys as soon as it joins a network. It shares a unique subset of keys with every other router in the network. Distance fraud can be prevented if each route that is sent from one router to its neighbor is attested by the advertising router's neighbor (the next hop on the advertised path) as well: When a router advertises a new route to a neighbor, it signs the update with each of its keys separately, sends all the signatures along with the update, and in addition also appends the set of signatures it received from its own neighbor from whom it learned the route. The router that finally receives the route has to validate both signatures (it shares a set of secret keys with both routers); if this verification is successful it can conclude that its neighbor did not advertise a wrong distance.

The overhead that this technique entails is in storage space for the secret keys that every router shares with every other router in the network and in computation time for distributing the keys and validating the signatures. According to the authors, this does not entail more overhead than authenticated RIPv2. In RIPv2 with MD5 authentication, keys must also be maintained and signatures must be validated. Thus, the complexity of both algorithms is similar. Moreover, RIPv2 does not offer corrective actions either, it also merely offers prevention and detection.

Summary: Tables 4.2 and 4.3 sum up the basic features of each approach and contrast advantages and disadvantages.

approach	features	advantages	disadvantages
RIPv2 with MD5 authentication	• Security Association determines the key that sender and receiver of an update use to encrypt a specific routing message • the key is appended to the message • message digest is computed and written into the authentication data field of the RIPv2 packet • receiver validates the message by computing the message digest and comparing it to the one it received • sequence numbers can be added	• easily implementable • low additional overhead • assures authenticity and integrity of a routing message • sequence numbers prevent from replay and deletion of messages	• a compromised legitimate node can still advertise false information • distances and next hops are not verified • no corrective actions
Implicit path method	• digital signatures and sequence numbers provide authenticity and integrity of routing messages • receiver of an update can verify a route using the new "predecessor" field and a loop-free path finding algorithm to recursively reconstruct the route	• protection against deletion, replay and modification of messages • protection against distance fraud • no transitive trust required • validates the complete route • works independently from network size	• cannot prevent from predecessor fraud • every router must have implemented the algorithm • relatively resource and time consuming
RIP-TP	• triangle theorem states that among a set of three nodes, the distance between one pair should be less than the sum of distances of the other two pairs • if the theorem does not hold: check between which two nodes the suspicious distance lies • sending of probing messages to the destination for verification of the update	• protects against distance fraud • incrementally deployable • expandable for large networks • a router can benefit from RIP-TP even if it is the only router in the network that has deployed RIP-TP • triangle theorem can be adjusted to other distance vector protocols besides RIP	• probing messages and replies can be manipulated: a malicious router can claim to be the destination router and send back a "port unreachable" message, or the probing message's time to live can be manipulated • no authentication mechanisms
PAIR	• helps to detect and recover from router attacks • instead of hop lengths, predecessor information and path sum metrics are transmitted in an update • a distance vector tree is built based on the predecessor information • metrics are updated and malicious updates can be detected based on these metrics • with the help of mathematical properties of path sum metrics inconsistency recovery can be done under certain circumstances	• offers the possibility of recovery in addition to detection of malicious updates • per node only 4 bytes of additional overhead are added in a packet • implementation is easy and complexity is low ($O(n)$)	• if more than one pair of entries per message is changed, there is no detection guarantee • conditions on which recovery is possible are rather restricted • no authentication mechanisms

Table 4.2: Comparison of approaches part 1

approach	features	advantages	disadvantages
S-RIP	• a router validates a new route by checking the consistency of that route with the nodes that propagated it • a reputation-based framework helps to decide how many routers to include in the consistency check • for authentication purposes, every router shares a different key with every other router in the network	• helps to prevent router impersonation, prefix impersonation and shorter/longer distance fraud • even if many nodes in a network are malicious, inconsistencies can be discovered if nodes do not collude • adjustable thresholds that help to balance security and efficiency • communication range of a node is dynamic	• if malicious nodes collude, wrong distances might remain undetected • detection of wrong distances is non-deterministic • consistency checks entail a large number of message transmissions • router prefix mapping requires a vast overview of the network topology • a node's reputation might be misinterpreted
S-DV	• based on S-RIP, but with modifications and improvements • trusted routers are introduced that limit the number of nodes which are involved in a consistency check • trusted routers cooperate to detect fraudulent distances • a predecessor attribute is added to each route • a security indicator is introduced that helps to choose secure routes over short but manipulated ones	• prevents router impersonation (using sequence numbers and MACs), prefix impersonation (using router prefix mapping) and shorter/longer distance fraud (using a distance request/reply mechanism) • distance request/reply mechanism reduces the number of messages considerably compared to S-RIP • detection of wrong distances is deterministic • compared to S-RIP a lot less keys are distributed and maintained	• who attests that trusted routers can really be trusted? • router prefix mapping requires a vast overview of the network topology
Symmetric key based techniques	• key distribution protocol assigns every router a set of symmetric keys • every router shares a unique subset of keys with every other router in the same network • when a router forwards a routing update it does not only add its own signatures (one for each key) but also the signatures of the router it has received the update from • the router that receives the update validates the signatures for which it shares the keys with the other two routers	• distance fraud can be prevented and detected • updates can be authenticated • low overhead • protects control plane and data plane	• fraud cannot be detected if neighbors collude (but if it is necessary the technique can be expanded so that more neighbors have to attest a distance; however, this entails more overhead)

Table 4.3: Comparison of approaches part 2

5 Conclusion and Outlook

In this thesis, different solutions to overcome security problems in distance vector routing algorithms were presented and compared. All of them induce extra overhead which is intelligible as additional information has to be stored in routing packets and further computations have to be executed. Besides providing authenticity and integrity of routing updates, the algorithms (with the exception of RIPv2 with MD5 authentication) are able to detect forged routing information on different conditions. Future work could try to relax these conditions so that detection is made easier. Algorithms that use cryptographic hash functions could use recent ones to provide a higher security level.

The authors of the examined papers plan to include the following in their future work: As shown in [PMZ03], triangle theorem checks can be generalized for use with other link metrics than hop-count and other routing protocols than distance vector routing protocols. Generalizing the verification of updates using probing messages is left to future work.

The authors of the pivot based algorithm for inconsistency recovery (PAIR) plan a prototype implementation of PAIR and also will try to improve PAIR in a way that it can detect false updates and recover under relaxed conditions.

As in S-RIP only individual fraud is considered and wrong distances might remain undetected if malicious nodes collude, the authors will try to find a solution for this shortcoming. Moreover, they will analyze S-RIP in detail and will compare it to other secure distance vector routing protocols. The basic ideas of S-RIP might even be used to secure BGP.

In ad hoc networks, when there are no trusted third parties, security and reliability are very important. This is why the authors of S-DV plan on extending their solution to routing protocols in ad hoc networks.

Bruhadeshwar et al. include the implementation and deployment of their symmetric key based approach in the internet in their future work.

Bibliography

[AF07] Atkinson, R. and Fanto, M.: *RFC 4822: RIPv2 Cryptographic Authentication*, 2007.

[BCBG06] Babakhouya, A., Challal, Y., Bouabdallah, M. and Gharout, S.: *SDV: A new approach to Secure Distance Vector routing protocols.* In *IEEE SecureCom / SECOVAL Workshop*, 2006.

[BKPD09] Bruhadeshwar, B., Kothapalli, K., Poornima, M. and Divya, M.: *Routing Protocol Security Using Symmetric Key Based Techniques.* In *ARES*, pages 193–200. IEEE Computer Society, 2009.

[CM03] Chakrabarti, A. and Manimaran, G.: *An Efficient Algorithm for Malicious Update Detection & Recovery in Distance Vector Protocols.* In *IEEE International Conference on Communications*, 2003.

[EG85] El Gamal, T.: *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms.* In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[fip09] *Digital Signature Standard (DSS).* Federal Information Processing Standards Publication, 2009.

[GLAM97] Garcia-Luna-Aceves, J.J. and Murthy, S.: *A Path-Finding Algorithm for Loop-Free Routing.* IEEE/ACM Transactions on Networking, 5:148–160, 1997.

[Gri09] Grimm, R.: *Grundlagen der IT-Sicherheit.* Lecture notes, University of Koblenz-Landau, 2009.

[Hed88] Hedrick, C. L.: *RFC 1058: Routing Information Protocol*, 1988.

[HPJ03] Hu, Y., Perrig, A. and Johnson, D.B.: *Efficient Security Mechanisms for Routing Protocols.* In *In Proc. NDSS'03*, pages 57–73, 2003.

[ius11] *The MD5 cryptographic hash function.* From Ius mentis – Law and technology explained: `http://www.iusmentis.com/technology/hashfunctions/md5/`, last accessed April 27th 2011.

[JK03] Jonsson, J. and Kaliski, B.: *RFC 3447: Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1*, 2003.

[Lin04] Lindqvist, J.: *Counting to Infinity.* Helsinki University of Technology, Seminar on Internetworking, 2004.

[Mal93] Malkin, G.: *RFC 1388: RIP Version 2 Carrying Additional Information*, 1993.

[Mal94] Malkin, G.: *RFC 1723: RIP Version 2 Carrying Additional Information*, 1994.

[Mal98] Malkin, G.: *RFC 2453: RIP Version 2*, 1998.

[Moy98] Moy, J.: *RFC 2328: OSPF Version 2*, 1998.

[ns211] *The Network Simulator – ns-2.* URL: `http://www.isi.edu/nsnam/ns/`, last accessed August 31st 2011.

[PD07] Peterson, L.L. and Davie, B.S.: *Computernetze: Eine systemorientierte Einführung.* dpunkt.verlag, Heidelberg, 4th edition, 2007.

[Per92] Perlman, R.: *Interconnections: Bridges and Routers.* Addison Wesley, Reading, Massachusetts, 1992.

[PMZ03] Pei, D., Massey, D. and Zhang. L.: *Detection of Invalid Routing Announcements in RIP Protocol.* In *IEEE Global Communications Conference (Globecom)*, 2003.

[Riv92] Rivest, R.: *RFC 1321: The MD5 Message-Digest Algorithm*, 1992.

[Smi97] Smith, B.R.: *Securing Distance-Vector Routing Protocols.* Master's thesis, University of California, Santa Cruz, 1997.

[Tan03] Tanenbaum, A.S.: *Computernetzwerke.* Pearson Studium, Munich, 4th edition, 2003.

[TNWZ00] Terzis, A., Nikoloudakis, K., Wang, L. and Zhang, L.: *IRL-Sim: A General Purpose Packet Level Network Simulator.* In *In Proceedings of the 33rd ACM-SIAM Symposium on Discrete Algorithms*, 2000.

[WKV03] Wan, T., Kranakis, E. and Van Oorschot, P.C.: *Secure Routing Protocols Using Consistency Checks and S-RIP.* Technical Report TR–03–09. School of Computer Science, Carleton University, Ottawa, Canada, 2003.

[WKV04] Wan, T., Kranakis, E. and Van Oorschot, P.C.: *S-RIP: A Secure Distance Vector Routing Protocol.* In *In Proc. of Applied Cryptography and Network Security*, pages 103–119, 2004.