Decentralized Security Mechanisms for Routing Protocols

by

Lakshminarayanan Subramanian

B.Tech. (Indian Institute of Technology, Madras) 1999
M.S. (University of California at Berkeley) 2002

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:
Professor Randy H. Katz, Co-Chair
Professor Ion Stoica, Co-Chair
Professor Scott Shenker
Professor John Chuang

Fall 2005

The dissertation of Lakshminarayanan Subramanian is approved:

_____

Co-Chair Date

_____

Co-Chair Date

_____

Date

_____

Date

University of California at Berkeley

Fall 2005

Decentralized Security Mechanisms for Routing Protocols

Lakshminarayanan Subramanian

Abstract

Decentralized Security Mechanisms for Routing Protocols

by

Lakshminarayanan Subramanian

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Randy H. Katz, Co-Chair

Professor Ion Stoica, Co-Chair

Today's Internet routing protocols are built upon the basic incorrect assumption that routers propagate truthful routing information. As a result, the entire Internet infrastructure is vulnerable to security attacks from routers that propagate incorrect routing information. In fact, a single router is capable of hijacking a significant fraction of routes by launching such an attack. This issue is not just restricted to Internet routing protocols but is widely prevalent in several routing protocols that have been proposed in the research literature. Many existing approaches for addressing the security problems of routing protocols typically assume the existence of a Public-Key Infrastructure (PKI) or some form of prior key distribution mechanism along with a central authority. While a PKI does enable addressing this security threat, building one such key-distribution infrastructure may not always be feasible. One faces serious deployment barriers in building an Internet-wide PKI with a central authority especially given that deploying one such architecture requires approval

across political and economic boundaries. Previous efforts for securing Internet routing and the Domain Name System using a PKI have not moved towards adoption.

In this dissertation, we address the following question: *Using purely decentralized mechanisms (void of a PKI and a central authority), what is the best level of security achievable for a routing protocol in the presence of adversaries?* One of the key conclusions that we arrive at is the direct relationship between decentralized security and the *reliable communication problem*. The reliable communication problem relates to determining the constraints under which a set of good nodes in a network can reliably communicate messages between themselves in the face of adversarial nodes in the network. We show theoretical results on the constraints under which the reliable communication problem is solvable. Based on these results, we describe the design of a *reliable communication toolkit* that implements our algorithms and provides a suite of generic security primitives that can be used to secure a variety of routing protocols. These security mechanisms supported by the toolkit are well suited for Internet routing since they are both easy to deploy as well as offer good security guarantees. We also show that the toolkit has broader applicability beyond routing protocols to: (a) achieve decentralized key distribution; (b) address the data integrity threat to the Domain Name System (DNS) in a decentralized manner.

Professor Randy H. Katz
Dissertation Committee Co-Chair

Professor Ion Stoica
Dissertation Committee Co-Chair

To my wonderful mom,

Contents

List of Figures

List of Tables

Acknowledgments

An Ode to the Three Elders: The Road to El Dorado

Walking in the wilderness, a cold dark stormy night,
The burning need to reach the doors of El Dorado by daylight,
A life at stake, hunger struck, death tapping at every door,
I was lost, scared, shivering, starving like never before.

After random wanderings, a pencil of light emerged from nowhere,
A ray of hope (I thought), in search for a way out of my despair.
Stumbled across woods, dirtied by the slush, with a rugged attire,
I ran like a mad dog to meet the Three Elders by the fire.

The Elders, knowledgeable of every stone in the wild,
The masters of the art of guiding even the tiny child,
Through the darkest of jungles, neglected by the moon;
A path to be discovered, I approach the Three Elders for a boon.

The road to El Dorado is not told but self-discovered, they say
Not a map exists with clues buried, the restless heart is at bay.
The impatience in my mind, the very thought of gold, should go,
Were their first words of wisdom, the same words that hit my alter-ego.

The day-dreamer of lofty expectations, that was I,
Needed lessons of reality from the Elders, to open my eye,
To see light in blackness, to hear sound in eternal silence,
To trace the road to El Dorado using my little lens.

The way is long, winding, often confusing, they remind,
To be steadfast despite adversity, something I had to find,
Preach Thou, the skill to look beyond into the far
And to not be lost, they point to the North Star.

With these new found priceless gems, I set out in complete poise,
Every being in the once gloomy jungle seemed all lit to rejoice,
To my sudden surprise, the path seemed all crystal clear
And I could discover the route to El Dorado without any fear.

The journey to El Dorado was long but not tiring,
For what I learnt enroute, every syllable had a deep meaning,
A meaning that will take ages to decipher, I think
Setting me in deep thought, till my eyes would wink.

An astounding heap of gold lay in front of me, like never before.
”Am I at El Dorado?”, I ask myself at its very door;
Momentarily blinded was I by the glitter of gold by daylight,
I left without a single piece, into the cold dark stormy night.

–Rivera (pen name of Lakshmi)

I dedicate this poem to my three amazing advisors Randy Katz, Ion Stoica and Scott Shenker, without whose guidance, my years at Berkeley would have been quite meaningless. During my graduate years, I consider myself extremely fortunate to have worked closely with an outstanding set of professors/researchers including Hari Balakrishnan at MIT, John Chuang at SIMS (on my dissertation committee), Mark Handley at UCL, Shivkumar Kalyanaraman at RPI, Venkata Padmanabhan at Microsoft Research, Jennifer Rexford at Princeton and Doug Tygar (qualifying exam committee chair). I have also had very fruitful research discussions with Tom Anderson, Eric Brewer, Sally Floyd, Mike Franklin, Ramesh Govindan, Joe Hellerstein, Anthony Joseph, Dick Karp, Subhash Khot, Adrian Perrig, Christos Papadimitrou, Kannan Ramachandran, Dawn Song, Satish Rao and David Wagner. I thank them all for their advice at various stages in my graduate school years. Its hard to forget Randy and his hiking trips where he is a mile ahead of the pack when everyone else is panting for breath, Randy gleefully telling me that I finished behind him at the Bay-to-breakers event, Randy scheduling a visit to his home (in between a 2 hr gap between flights) to sign my MS thesis, proclaiming to Scott that he does not like chocolates (at the qualifying exam), scheduling Scott for 10 continuous ICSI talks to present his 10 Sigcomm submissions (ask Scott for his famous response to this!), the frantic calls to Ion in the middle of my job interviews asking for tips, my vain attempts trying to convince George (Ion's son) that I performed a magic trick that made ice in my hand disappear (the ice actually melted), Randy and his jokes when we were in Finland (my god Randy at his best!), my strong reactions expressed to Hari and Ion on the OverQoS paper rejection from Sigcomm (that was the lowest point in my PhD!), Scott asking me to stand up (and make a fool of myself) in his EE122 class when I sneaked in (to see how he teaches), Ion momentarily disowning me as a student at Sigcomm'01 (for fun of course!) and asking others whether they wanted

to take me as a student, me pacing nervously outside two professors' offices before I met them first (later I figured that their names where Randy and Scott).

To my awesome friends

Life in Berkeley would have been no fun without friends, be it at work, home or play. From the research side, I have had the privilege to work and interact with a brilliant set of students who have taught me a lot. Volker Roth contributed a lot as a collaborator in the BGP security and the reliable communication theory chapters. Our regular meetings at Strada and I-house cafe contributed much towards this thesis. Sriram Sankararaman worked with me on the reliable communication toolkit described in this thesis. Sriram and Venu Ramasubramanian (Cornell) have contributed towards the ideas presented in the DNS chapter. The other people who have contributed a lot include (in no particular order): Karthik Lakshminarayanan, Ananth Rao, Brighten Godfrey, Dilip Joseph, Jayanthkumar Kannan, Shelley Zhuang (DFJ), Sharad Agarwal (MSR), Helen Wang (MSR), Chen Nee Chuah (UC Davis), Sridhar Machiraju, Mukund Seshadri, Cheng Tien Ee, Matthew Caesar, George Porter, Bhaskaran Raman (IIT-Kanpur), Nick Feamster (MIT), David Andersen (CMU), Weidong Cui, Fang Yu, Ana Sanz Merino (Ericsson, Spain), Li Yin, Sylvia Ratnasamy (Intel), Rabin Patra, Melissa Ho, Sergiu Nedevschi, Sonesh Surana, Rodrigo Fonseca, Dennis Geels, Morley Mao, Wilson So, Yong Xia (NEC, China), Amoolya Singh, Ramakrishna Gummadi, Tina Wong (CMU), Amol Deshpande (Maryland), Vijayshankar Raman (IBM Almaden), Ben Zhao (UCSB), Ratul Mahajan (Washington), Krishna Gummadi (Washington), Emre Kiciman (Stanford), Mubaraq Mishra, Mel Tsai, Rahul Shah (Intel), Kiran (Qualcomm), Antar Bandyopadhyay (Chalmers, Sweden), Arnab Nilim (Citigroup), Rajarshi Gupta (Qualcomm) and Sunil Bhave (Cornell). I would like thank them all. I owe special thanks to all the students who have attended my practice talks,

read early versions of papers and provided amazing feedback.

My life at Berkeley has been great fun. It all starts with regular parties at home thanks to an awesome set of room-mates (over six years): Mubaraq, Rahul, Ali, Kiran, Ashwin, Mudit, Ganesh and Gopal. Several fun moments at Berkeley are worth mentioning: the Bengali-filled hangouts at home (Antar, RG, Nilim, Rumana, Anindita, Dickey, Rahul, Kiran, Arindam, Manju, Thathagat, Krishnendu, Rupak, Tandra, Gaurav), regular dancing trips to the city, regular visits to pricey restaurants, my forays into fantasy baseball (thanks to Sunil), the innumerable squash games at the gym (particularly with Sunil, Mubaraq, Sushil, Sriram, Jayanth, Sonesh, Sushant), my valiant attempts with weights and running, guitar jamming sessions, music recordings, late night movie sessions, innumerable junk hindi movie rentals. Life at 465 Soda has been fun thanks to Sylvia, Wilson, Shelley and Karthik. Life at Intel the past few months has been great thanks to the company of Rabin, Melissa, Sergiu, Rosenblum, Sonesh, Joyojeet, Bowei, Demmer, RJ and them being immune to my random music recordings late at night (without which this thesis would not have been done). Coffee and guitar now form an integral part of my life. Strada, FSM, Brewed Awakening, Starbucks and Nefeli cafes have profited a lot from my coffee addiction (100$ a month) except the free Espresso machine at Intel; without them, research would not have been possible. And, how can I forget the additional revenue I generated for them by dragging my friends (whoever is free at that time) everyday to these shops. Thanks to my roommates Rahul and Mubaraq, I made a truck load of awesome friends in BWRC: Jana, Delynn, Maryam, Hoyos. My friends from Asha are very hard to forget: Anand, Bhaskar, Shashi, JJ, Akanksha, Neha, Amrita, Vishy, Charu, Sandipan, Parag. I also spent a lot of time at ICSI where I made some of the funkiest friends (foozball rocks man!): Chema, Funky cold Medina, Christian, Pedro, Abhishek, Sarang, Nick (will miss his questions!), Robin, Jerry, Jaeyeon, Sylvia

(I am following her path: Berkeley $->$ ICSI $->$ Intel; logically she should come to NYU next!), Walfish, Arun, Ratul, Atanu Ghosh, Orion (one British guy who actually plays squash), Pavlin, Eddie (everyone at ICSI knew him), Jitu, Leah, Maria, Jaclyn, Aditya, Krishna, Holger. Other friends who have had a big impact on me at Berkeley include Kannan Viswanath, Jayanth Krishnan, Vijay Parthasarathy, Anand Ganesh, Devavrat Shah, Arvind Arasu, Shivnath Babu, Hemang Kapasi, Venu Ramasubramanian, Sirish Chandrasekaran and Sandip Das (my unofficial guitar teacher!). After all these names, I am sure I forgot a few. I thank my friends with all my heart for all that they have done for me.

To my family

This entire process of graduate school would absolutely have not been possible without the support of my family. To me, my mom (Mangala) is by far the greatest personality that I have come across; the hardships she has been through to raise me after my dad expired when I was six have been phenomenal. I owe this thesis to her. My sister (Srilakshmi) has been my best friend; anything and everything, I chat with her. My uncle (Viswanathan) and my grandfather (Vaidyanathan) have been pillars of support for my family ever since my dad expired. I also owe my gratitude to my close cousins Rajeswari, Krishnan, Usha, my uncle Kothandaramaswamy and my grandmom Pattammal. I wholeheartedly would like to thank them all for their support.

Chapter 1

Introduction

“ Lord Krishna says to Arjuna: ’He who practices the truth without knowing the difference between truth and lies is a fool.’ ”

– Book of Karna, Mahabharatha

Winston Churchill once jokingly said, *“ A lie gets halfway around the world before the truth even has a chance to get its pants on”*. What was once meant as a joke, now sadly stands as the raw truth about the current state of affairs of Internet routing.

Today, a single malicious router can hijack a large fraction of Internet routes by merely propagating incorrect routing information. For example, in 1997, a simple configuration error in a router caused it to advertise incorrect routes claiming direct connectivity to several destinations. This resulted in a massive black hole that disconnected significant portions of the Internet [29, 84]. Similar Internet-wide disruptions resulting from configuration errors have been documented in the past [77, 79]. Router configuration errors occur on a very regular basis, affecting roughly a few hundred

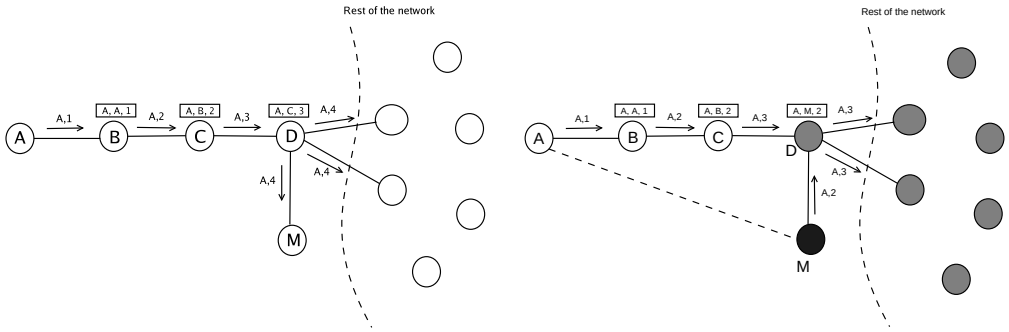
destinations everyday [79].

This dissertation is centered around the question: *" How do we protect the Internet from routers that lie?"*. This question is relevant because current Internet routing protocols are built upon the basic assumption that routers propagate truthful information. Yet, there is nothing in existing routing protocols that ensures this assumption holds. As a result, the entire Internet infrastructure is vulnerable to security attacks from routers that propagate incorrect routing information.

This issue is not restricted to the Internet; the security threat posed by incorrect routing advertisements from malicious nodes is prevalent in several other routing protocols proposed in the literature [36, 35, 100, 64, 61, 92] and deployed in other real-world networks [2, 6, 14, 10]. In all these routing protocols, a single malicious node is sufficient to disrupt the routing process in a large fraction of the nodes within the network. To better understand this phenomenon, we now describe a simple example network where a malicious router can hijack several routes using a single incorrect routing advertisement.

1.1 A simple example

We begin by providing a basic idea of how a routing protocol works. A routing protocol is a distributed process by which nodes within a network exchange reachability information and establish communication paths with other nodes in the network. In most routing protocols, a node is initially only aware of its neighbors and discovers routes to other nodes by exchanging *routing messages* with its neighbors. In the process, every node establishes a *routing table* that contains a next-hop entry corresponding to other nodes in the network.



(a) (b)

Figure 1.1: A simple example where a single malicious node M can disrupt several routes to node A by propagating a single incorrect routing announcement. Part (a) depicts the execution of the distance vector routing protocol and the propagation of routing messages. In part (b), the grey-colored nodes represent the set of nodes affected by the incorrect announcement.

Consider the topology described in Figure 1.1(a) where all nodes choose their next hop neighbors based on the shortest path calculated using the distance-vector routing protocol. In distance vector routing, each node exchanges reachability information about other nodes in the network along with the shortest path length to each node. In the example topology illustrated in Figure 1.1(a), C establishes a route to A based on B's routing advertisement and further propagates the advertisement to D. The remaining nodes in the network establish a route to A through D.

Consider the case where M acts as a malicious node and propagates a single bogus announcement claiming direct connectivity to A (as illustrated in Figure 1.1(b)). Since the path-length through M appears to be shorter, D ends up choosing the route through M (instead of the genuine route through C). Once M has inserted itself along the routing path to A, it can *drop* all the packets thereby rendering A to be unreachable to D. In the process, all the nodes that connect to A through D also get affected by this incorrect announcement. This example illustrates the fact that *a single incorrect routing announcement can potentially affect several routes to a destination.*

1.2 Problems with Internet routing

Akin to the example illustrated in the last section, Internet routing suffers from the same security threat of incorrect routing advertisements, but at a much larger scale. The Internet is a collection of Autonomous Systems (AS), numbering more than 17,000 in a recent count in early 2005 [125, 107]. The Border Gateway Protocol (BGP) [116], the current inter-domain routing protocol, knits these autonomous systems together into a coherent whole. BGP currently enables routers to transmit route announcements over authenticated channels, so that malicious nodes cannot impersonate the legitimate sender of a route announcement. This approach, which verifies *who* is speaking but not *what* they say, leaves the current infrastructure extremely vulnerable to both configuration errors and deliberate attacks.

To recap the 1997 incident [29], a single router of an autonomous system simultaneously propagated over a thousand incorrect routing announcements claiming direct connectivity to a large number of different destinations. Routers in other autonomous systems blindly believed these routing advertisements to be correct and many of them ended up choosing the newly advertised route (since it appeared shorter) over the previously existing route, thereby rendering several destinations unreachable. This incident resulted in a massive outage that brought down the Internet for several hours. It took the collective effort of several network operators in different autonomous systems lasting many hours to manually trace the location of the problem and fix it.

1.2.1 Seriousness of the problem

We illustrate the seriousness of the problem along two dimensions. First, incorrect routing announcements are very widespread in the Internet routing protocols today making this threat a very real one. Second, in the worst case, a single randomly-placed malicious router can hijack *over a third* of the Internet routes by generating incorrect announcements. We will now elaborate on these two aspects.

Source of the problem: Configuration errors and router compromises by attackers are two of the primary reasons for a router to propagate incorrect routing information. Configuration errors due to buggy scripts and human errors are a very common occurrence and generate incorrect routing advertisements for roughly $200 - 1200$ destinations daily [79]. This represents a very noticeable fraction ($0.2 - 1\%$) of entries in a typical Internet routing table. Apart from configuration errors, router compromises represent a much more serious threat. Routers are surprisingly vulnerable to compromises. Several routers are known to still have *default passwords* [15, 122] set by the vendor and a customer who purchases a router from a vendor may forget to reconfigure the default password. An attacker who is aware of the default password can trivially compromise the router. Additionally, routers use standard interfaces like telnet and SSH, which have their own share of software vulnerabilities. From our perspective, the primary differences between configuration errors and router compromises are two-fold. First, attackers who compromise routers can take active countermeasures to the security mechanisms we may build while misconfigured routers do not. Second, an attacker can compromise multiple routers and use them as a set of *colluding* adversarial nodes that together launch a targeted attack.

Worst-case attack scenario: In order to maximize the number of hijacked routes, a misbehaving router can advertise incorrect routing announcements claiming direct connectivity to every destination. Conceptually, this can be considered a generalized version of the 1997 router misconfiguration incident. By claiming direct connectivity to a destination, the path-length of the incorrect route in many cases tends to be shorter than the path-length of the actual route to the destination, thereby forcing other nodes to choose the incorrect route. Based on analyzing the structure of the Internet topology over the past 3 years ($2002 - 2005$) gathered from several routing data sources [125, 107], we found that a single randomly placed router can hijack *over one-third* of Internet routes in the median case. The exact fraction of routes that can be hijacked by a single router is dependent on the location of the router in the topology. There are many well-placed routers in the Internet (located in tier-1 autonomous systems) which can hijack *over 70%* of routes using this attack.

This clearly shows that a single router is sufficient to cripple the Internet by hijacking a substantial fraction of Internet routes. The reasons as to why this fraction can be so high are two-fold. First, in a routing protocol, every node is initially unaware of the network topology thereby making it easy for an attacker to claim new edges to be existing in the topology. There is no simple way of verifying the connectivity claims of a node. Second and more importantly, a single malicious router within an AS can internally (within the AS) propagate incorrect routing information and corrupt the routing state of all the routers within an AS thereby making the entire AS act in a malicious manner. Inter-domain routing using BGP treats every AS as a single coherent entity. This makes a single malicious router within an AS equivalent to the case of the entire AS being malicious. The power of an adversary substantially increases when the attacker uses a set of compromised routers in different ASs to simultaneously replicate the same attack from different locations.

These two arguments are used to drive home the point that the threat of incorrect routing advertisements is a very important one and, if left unaddressed, can have serious implications on the security of the entire Internet infrastructure.

1.3 Secure routing problem setting

In this section, we provide a brief setting of the secure routing problem along with the threat model in order to establish the context for describing the key results presented in this dissertation. We provide a more detailed description of the secure routing problem in Chapter 2. In this section, we first distinguish between two dimensions of the secure routing problem – one that occurs at the control-plane and the other at the data-plane. Later, we describe the threat model.

1.3.1 Control plane vs Data plane security

Every routing protocol is associated with a *control plane* and a *data plane*. The control plane of a routing protocol deals with the process by which routers exchange routing information and establish routes. The data plane deals with the actual routing of data packets once the routes are established in the control plane. In order to secure any routing protocol in the face of misbehaving nodes, it is essential to solve two problems:

Control plane security: A misbehaving node can *hijack* a route when it propagates an incorrect routing advertisement causing other nodes to choose the incorrect route over other routes. Control-plane security deals with verifying the validity of routing announcements propagated by nodes.

Data plane security: A misbehaving node that is along the routing path to a destination can create data-plane problems by dropping packets or forwarding packets in a manner inconsistent with the routing advertisements it has received or propagated. Data plane security deals with checking whether the underlying routes set up in the control plane deliver data packets to the corresponding destinations.

Control plane security and data plane security are two separate problems and we require separate mechanisms to address these problems. In the context of Internet routing, Mao *et al.* [81] show that for nearly 8% of Internet paths, the control plane and data plane paths do not match. Whenever the routes in the control and data plane have a mismatch, verifying the correctness of the control plane does not provide any guarantees on the correctness of the route in the data plane. The fact that a sizable fraction of Internet routes have a control-data plane mismatch motivates the need for separately verifying the correctness of routes in the data plane and not merely focusing on the control plane. Prior work [70, 72, 112, 92, 65, 114, 57] on securing Internet routing focuses primarily on the control plane.

Control plane misbehavior can be triggered by configuration errors or deliberate attacks. Data plane misbehavior, on the other hand, can be triggered either due to deliberate attacks (malice), forwarding errors in routers or due to genuine reasons triggered by a control-data plane mismatch (*e.g.,* a failure along a routing path in the data plane that may not manifest in the control plane due to a control-data plane mismatch).

1.3.2 Adversary model

Any misbehaving node, be it in the control plane or in the data plane, is treated as an adversarial node. We define three classes of adversaries in our system: *isolated adversary*, *independent adversaries* and *colluding adversaries*. An isolated adversary covers the case of a single compromised node or a single node with a configuration error. Independent adversaries refers to a collection of isolated adversaries that act independently. A collection of different routers each with its own set of configuration errors can be treated as a set of independent adversaries. Colluding adversaries, in comparison to independent adversaries, have the additional ability to tunnel and share common information between them during protocol execution in order to potentially thwart security defense mechanisms. Colluding adversaries can also fake the existence of direct links between them in the topology.

From the control-plane perspective, an adversarial node can: (a) drop routing messages; (b) generate spurious routing messages; or (c) modify routing messages. From the data-plane perspective, an adversarial node can: (a) drop data packets traversing the node or (b) misroute packets to different neighbors not along the routing path to the destination.

1.4 The need for a decentralized security solution

Many existing approaches [100, 32, 114, 92, 72, 112, 65, 61] for addressing the security problems of routing protocols typically assume the existence of a public-key infrastructure (PKI) or some form of prior key distribution mechanism along with a central authority to enable node authentication. This is specifically true of several proposals [72, 112, 65, 92] to secure Internet routing that make

extensive use of digital signatures and public key certification. The *Secure-BGP (S-BGP)* [72, 70] and the *Secure Origin BGP (soBGP)* [112] are two of these proposals which have working groups in the IETF [4], the body that governs the Internet protocol standards.

Verification of routing information in Secure-BGP using a PKI works as follows. Secure-BGP appends each routing advertisement with a signature generated using its public-key. Any router can verify the validity of the signature of every autonomous system along a path using the PKI since the public key of every autonomous system is derived from the central authority. Hence, PKI provides a trusted communication channel between every pair of autonomous systems through a central authority.

While the presence of a PKI does enable addressing many of the security issues, building one such key-distribution infrastructure may not always be feasible. Many of the prior security proposals for securing the Internet architecture (not just routing) face a serious deployment barrier since they typically require an extensive cryptographic key distribution infrastructure and a trusted central authority. Neither of these two ingredients is currently available, nor is it likely that they will be in the near future since building an Internet-wide Public Key Infrastructure (PKI) requires approval across political and economic boundaries.

Deploying an Internet-wide PKI is a very difficult task due to several reasons including technical, political and economic ones. PKIs impose a heavy technological and management burden [43, 49, 41]. For a PKI model to be successful, it would require all the autonomous systems to adopt the standard and fall under the purview of a single central authority. Given that several autonomous systems are within specific countries, the biggest political hurdle to PKI acceptability for Internet routing is that several countries may oppose the very model of their national telecommuni-

cations networks to be even partially under the control (or even answerable) to a central authority outside the country. One striking example of this political dimension occurred in the recent UN Internet summit at Tunis [16] where several governments had a long political argument about whether ICANN (the body that is responsible for allocating IP addresses) should be held answerable to the US government or to the UN. Apart from the political aspect, building an Internet wide PKI infrastructure incurs huge costs and has a high risk of failure thereby posing a very high barrier to entry. Due to this deployment hurdle, the Secure-BGP [72], soBGP [112] and Secure-DNS [44] efforts have not moved towards adoption. Secure-BGP has been adopted only by a very small number of ISPs after 5 years since its introduction.

This motivates the need for developing *decentralized* security mechanisms that are both *easy to deploy* as well as provide *good security guarantees*. These security mechanisms should also be completely void of both a public-key infrastructure and a trusted authority. In the Internet, deployability and good security are often at odds with each other and simultaneously achieving both is a grand challenge. Conventional wisdom has been that a key distribution mechanism is essential to address many of the security threats, especially since key distribution automatically provides node authentication. Node authentication becomes much harder to establish in a purely decentralized manner without a trusted authority.

1.5 Main contribution

The problem of developing decentralized security mechanisms for routing protocols has long been an open and elusive research challenge. The primary contribution of this dissertation is to address

this research challenge. More specifically, we answer the following question: *Using purely decentralized mechanisms, what is the best level of security achievable for a routing protocol in the presence of adversaries?*

In this dissertation, we have developed a suite of decentralized security mechanisms that are generic and can be used to secure a variety of routing protocols. These security mechanisms are well suited for Internet routing since they are both easy to deploy as well as offer good security guarantees. Additionally, they do not rely on any prior key distribution mechanism or a central authority. The results that we establish in this work provide a foundational set of building blocks that are potentially applicable for achieving decentralized security in other distributed systems beyond routing protocols. Specifically, we show how our techniques can be extended to secure the Domain Name System (DNS) [86, 87] in a decentralized manner.

The larger vision behind this dissertation is to determine the basic set of principles essential for building decentralized security solutions for distributed systems. In the process, we seek to understand the strengths and limitations of adopting a decentralized approach towards security. One of the conclusions that we arrive in this dissertation is the direct correspondence between decentralized security and the *reliable communication problem* [22, 23, 45]. The reliable communication problem relates to determining the constraints under which a set of good nodes in a network can reliably communicate messages between themselves in the face of adversarial nodes in the network. This problem arises in the context of the classic Byzantine agreement problem in distributed systems [98, 76]. To achieve decentralized security, it is essential to establish a *reliable communication channel* between every pair of good nodes that enables every node to reliably communicate messages to other nodes in the face of adversaries attempting to disrupt the communication. By

addressing the reliable communication problem, we establish two basic constraints that must be satisfied by any distributed system in order to achieve decentralized security:

- **Connectivity constraint:** The underlying network that interconnects a set of nodes should have a minimum vertex connectivity of $2k + 1$ in order to achieve reliable communication in the face of k colluding adversaries. The vertex connectivity of a network is the minimum number of vertex-disjoint paths between any pair of nodes.

- **Fixed-identity constraint:** An adversarial node that uses multiple identities when communicating with different nodes is often referred to as a *Sybil attacker* [47]. We show that one cannot achieve reliable communication in the face of *Sybil attackers* [37, 119] irrespective of the connectivity of the underlying network. In other words, every node in the network is required to have a unique identity that it cannot fake to its neighbors.

These two constraints have important implications on the applicability of decentralized security for a variety of distributed systems. The connectivity constraint illustrates the fact that "perfect" decentralized security is feasible only in the face of a *bounded set of adversaries*. A centralized approach using a public-key infrastructure is better suited to handle a larger number of adversaries since the PKI provides a trusted channel between every pair of nodes using a central authority. The fixed-identity constraint states that decentralized security is impossible to achieve in distributed systems where nodes do not have fixed identities. Mobile ad-hoc networks [55] and dynamic peer-to-peer networks [94] are two example networks where decentralized security is very hard to achieve since an attacker can potentially exhibit Sybil behavior in these networks. We will now briefly elaborate on the need for these two constraints for decentralized security.

While these two constraints might seem stringent at the onset, they are both essential as well as achievable in practice. The Internet as well as many other networks that we operate in today have some form of fixed identities. For example, in Internet routing, every AS is associated with a fixed identity in the form of unique AS number which is known to all its neighbors. The connectivity constraint does impose a theoretical bound on the number of adversaries that a decentralized security mechanism can handle. However, in many networks, our primary design goal is to defend against only a few adversaries; a misconfigured router is equivalent to the case of a single adversary. Within the well-connected core of the Internet comprising tier-1 and tier-2 ISPs, we can achieve decentralized security in the face of up to 10 adversaries but not more. Additionally, in the face of several adversaries in the system, even a PKI based approach does not provide strong security guarantees. Therefore, it becomes fundamentally hard to provide any form of security guarantees in the face of several adversaries in the system.

Fixed Identities vs Public Key Infrastructure: Finally, we reinforce the fact that satisfying the fixed identity criterion is much easier than building an infrastructure-wide PKI. The reasons are three-fold. First, if fixed-identities exist, then it is easily enforcible in the face of misconfigurations and node compromises. Second, with the advent of different forms of tamper-proof hardware [124], many devices come with inbuilt fixed-identifiers. Third, enforcing the fixed-identity constraint requires only local coordination, which is much easier to enforce than performing system-wide coordination to build a full-fledged PKI. With multiple administrative domains, one faces deployment hurdles for system-wide coordination [43, 49].

1.6 Research progress path and thesis organization

Our research progress on the problem of building decentralized security mechanisms for routing protocols can be categorized into four stages as illustrated in Figure 1.2. In the first stage, we studied the Border Gateway Protocol (BGP) as a specific case of a routing protocol to secure in a decentralized manner. In the second stage, we extend the techniques developed for BGP for other routing protocols. Here, we establish the connection between decentralized secure routing and the reliable communication problem. In the third stage, we describe the implementation of a practical toolkit that exports generic security primitives for different routing protocols based on the theoretical techniques developed to address the reliable communication problem. In the fourth stage, we show the applicability of the toolkit beyond routing protocols; here, we demonstrate how the reliable communication toolkit can be used to address the data integrity security threat to the Domain Name System (DNS).

For ease of exposition, we organize the thesis in the same order as the natural progression represented by the four stages. In Chapter 2, we set the basic background by presenting an overview of the secure routing problem and describing how this problem relates to the reliable communication problem. In this chapter, we formulate the secure routing problem both from the perspective of the control-plane and data-plane and elaborate the challenges that we face in addressing them.

The next five chapters (Chapter 3 - Chapter 7), describes each of the different research stages in detail. Now, we will briefly walk the reader through our thought process behind the various research stages in our methodology. We will also briefly summarize the salient aspects of each of the research stages.

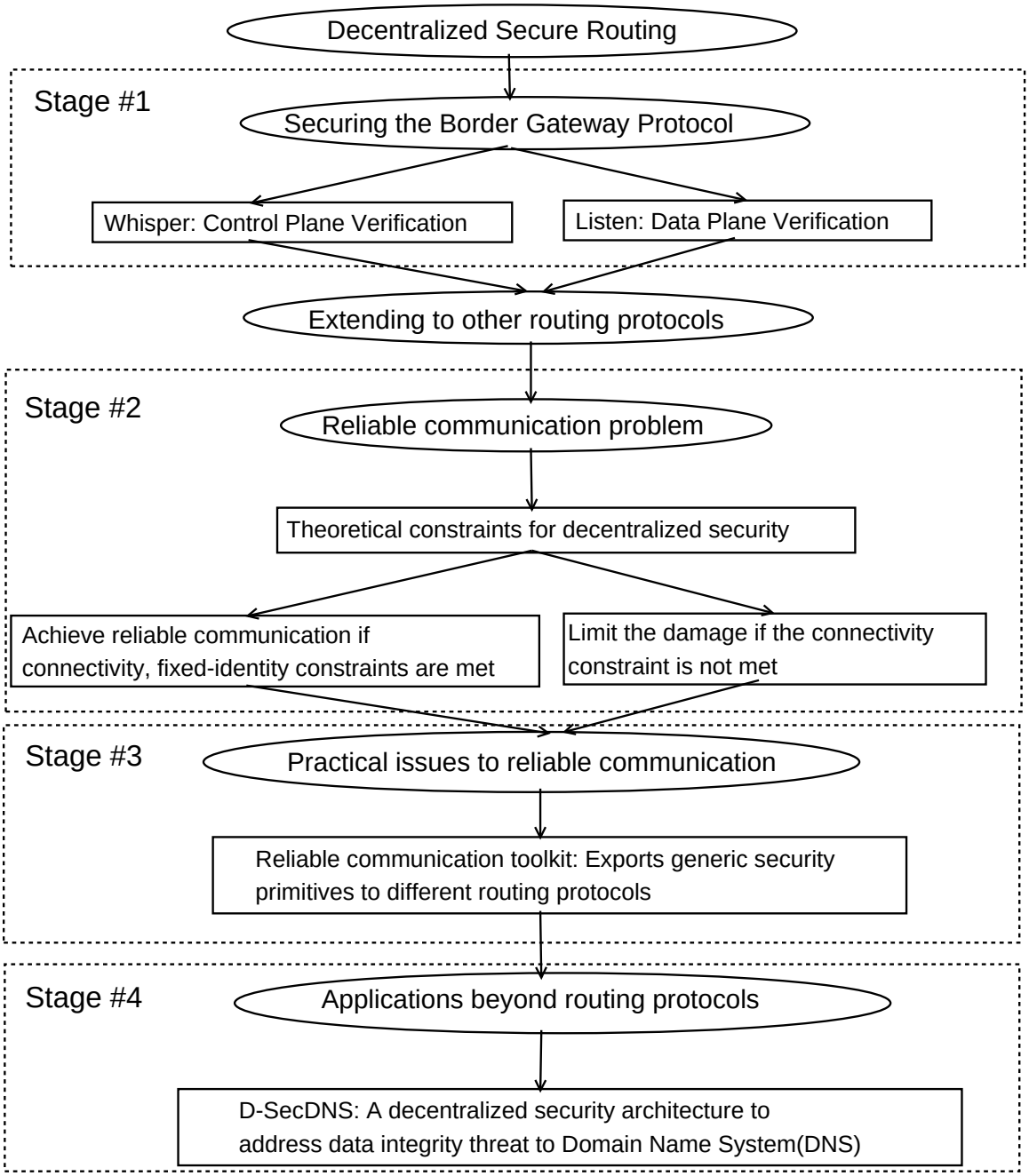


Figure 1.2: Research progress path

1.6.1 Stage #1: Securing the Border Gateway Protocol

Our exploration into the space of decentralized security, began with a search for an answer to the following question: *How can we secure the Border Gateway Protocol (BGP) in a decentralized manner?* BGP is the current inter-domain routing protocol used in the Internet. Prior proposals for improving BGP security relied on a public-key infrastructure. Hence, BGP was a natural first choice as a case study for the secure routing problem.

To address the security problems with BGP, we propose a combination of two mechanisms: *Listen* and *Whisper*. Listen passively probes the data plane and checks whether the underlying routes to different destinations work. Whisper uses cryptographic functions along with routing redundancy to detect bogus route advertisements in the control plane. These mechanisms are easily deployable, and do not rely on either a public key infrastructure or a central authority like ICANN. The combination of Listen and Whisper eliminates a large number of problems due to router misconfigurations, and restricts (though not eliminates) the damage that deliberate attackers can cause. In Chapter 3, we provide a detailed description of Listen and Whisper.

1.6.2 Stage #2: Reliable communication problem

Based on our experience developing security mechanisms for BGP, the next natural question that arises is: *What basic primitives are essential to achieve decentralized security in routing protocols other than BGP? Are these primitives dependent or independent of the underlying protocol?*

In order to address this question, we need to first theoretically formulate the secure routing problem and outline the exact set of security guarantees that we require for each routing protocol. In

Chapter 2, we define a protocol independent version of the secure routing problem. Based on this problem formulation, we establish a direct correspondence between *secure routing* and the *reliable communication problem*. Addressing the reliable communication problem provides a trusted channel between good nodes which is a fundamental building block to building secure routing protocols.

In Chapter 4, we describe our theoretical results on the solvability of the reliable communication problem. We show that the reliable communication problem can be solved if and only if the connectivity and fixed-identity constraints are satisfied. We also show that the fixed-connectivity constraint is fundamentally essential in that if it is not met, then a single adversary is sufficient to disrupt reliable communication irrespective of the connectivity of the network.

In Chapter 5, we study a variant of the reliable communication problem in *sparse networks* which do *not* satisfy the connectivity constraint. In sparse networks, it is fundamentally impossible to achieve reliable communication between every pair of nodes. Hence, decentralized security is not achievable *i.e.,* malicious adversaries can potentially hijack a certain fraction of routes. In such networks, we show that it is possible to *limit the damage* that adversaries may cause in sparse networks. We specifically study the reliable communication problem for the case of a single adversary ($k = 1$) and prove optimality results for this case. Our techniques are also applicable to the case of multiple adversaries $(k > 1)$. However, determining the optimal defense strategy for $k > 1$ is still an open research problem.

1.6.3 Stage #3: Generalizing to other routing protocols

During the previous research stage, we primarily established the basic theoretical techniques and algorithms essential to achieve reliable communication. This only addresses a portion of the decen-

tralized secure routing problem. Two questions remain to be addressed: *(a) How do we leverage reliable communication to secure different forms of routing protocols? (b) Are these techniques feasible and practical to implement?*

To address these two questions, we developed a *reliable communication toolkit* that implements the basic algorithms to achieve reliable communication. This toolkit provides a generic set of security primitives that can be appended to different routing protocols and secures them. The primitives exported by the toolkit are independent of the routing protocol. We illustrate the applicability and the generality of the toolkit by integrating it with three standard types of routing protocols, namely: link-state, path-vector and distance-vector routing. Based on a detailed performance evaluation of the toolkit, we show that decentralized security is *feasible, practical and not expensive* under the assumption that the number of adversarial nodes is small. We elaborate on the design of the toolkit in Chapter 6.

1.6.4 Stage #4: Applying reliable communication to the Domain Name System

The reliable communication toolkit has broader applicability beyond routing protocols, especially for providing decentralized key distribution (since nodes can reliably communicate their public keys to other nodes). Using this toolkit, one can translate existing PKI-based solutions to purely decentralized solutions provided reliable communication is achievable.

In Chapter 7, we describe how the reliable communication concept can be applied to address the data integrity problems associated with the Domain Name System (DNS) [86, 87, 89]. The DNS suffers from data integrity threats that arise due to configuration errors or malicious adversaries compromising name servers and propagating incorrect DNS responses to end-host queries [74, 44, 96]. The

DNS delegation process introduces *transitive trust* relationships between different domains which further worsens the data integrity problem [104, 123]. The delegation process introduces complex dependencies in the system which can enable a single compromised DNS server to affect not only its domain but also several other domains outside of its control. All of the prior security proposals [44, 48, 19] for addressing the data integrity problem of the DNS rely on a public key infrastructure with a central authority. In Chapter 7, we describe the design of D-SecDNS, a decentralized security architecture for the DNS that addresses the data integrity threat to the DNS.

Finally, in Chapter 8, we present our conclusions where we revisit the key contributions of this dissertation and discuss potential avenues for future directions.

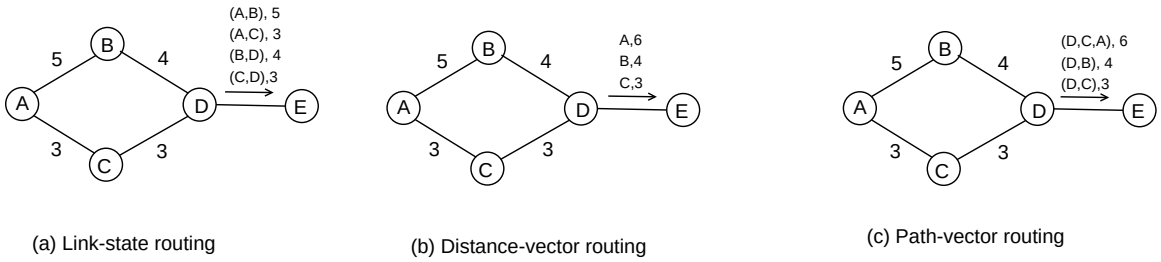
Chapter 2

Secure Routing Problem

“ Once we roared like lions for liberty; now we bleat like sheep for security. ”

– Norman Vincent Peale, American writer

In this chapter, we will describe a theoretical formulation of the secure routing problem and describe how this problem relates to the reliable communication problem. In Section 2.1, we begin with a brief introduction about the control and the data plane of a routing protocol. In Section 2.2, we illustrate the differences between securing the control plane and the data plane of routing and why these represent two separate problems. In Section 2.3, we define the secure routing problem both from the control plane and the data plane perspective. Next, in Section 2.4, we describe the reliable communication problem and show its correspondence to the secure routing problem. Here, we describe why prior work on reliable communication is insufficient to address the secure routing problem and describe a variant of the original reliable communication problem that needs to be addressed in order to achieve secure routing. In Section 2.5, we provide a summary of our



(a) Link-state routing (b) Distance-vector routing (c) Path-vector routing

Figure 2.1: Types of routing messages for three standard types of routing protocols: link-state, distance-vector and path-vector routing. The type of routing messages is indicated along the link (D, E) where D propagates routing messages to E.

theoretical results on the reliable communication problem. Finally in Section 2.6, we describe prior work on secure routing which have largely focused on using some form of prior key distribution. In Section 2.7, we summarize the conclusions from this chapter.

2.1 Routing protocols: A brief overview

The *control plane* of routing deals with the process by which nodes within a network exchange routing information and establish routes to other destinations. The *routing protocol* specifies the manner in which nodes exchange routing messages and compute routes to different destinations. The *data plane* deals with the forwarding of data packets once the routes are established in the control plane.

The type and structure of the routing messages exchanged between nodes is dependent on the underlying routing protocol. Every routing protocol is also associated with a *route selection* process which defines the routing metric used for choosing the best route to every destination. The simplest routing metric used by many routing protocols is the *shortest-path* metric, in which every node chooses the route with the shortest path-length to every destination. In a more general version of

shortest-path routing, every link is associated with a cost value and every node chooses the route with the lowest net-cost to each destination. Apart from shortest-path routing, there are several other forms of routing metrics. As an example, in Chapter 3, we will describe the Border Gateway Protocol (BGP), the current inter-domain routing protocol, which uses *policy-based* routing where every node can use its own local policies for route selection.

Figure 2.1 illustrates the type of routing messages for three standard types of routing protocols: link-state routing, distance-vector routing and path-vector routing. The basic form of all three protocols use the shortest path metric for route selection. In link-state routing, every node learns the entire network topology along with the link-costs and computes the shortest path to each destination. In link-state routing, every routing message contains information regarding a specific link or a set of links in the network. In distance-vector routing, every routing message propagated by a node to its neighbors contains the identity of a destination and the length of the shortest path learnt to that destination. Path-vector routing is a variant of distance-vector routing, where every routing message contains the entire path comprising of the identities of all nodes along the route to each destination. BGP uses path-vector routing for propagating routing messages.

Using the routing protocol, every node computes a *routing table* which stores the next-hop in the routing path to each destination. In the data-plane, every node uses the routing table to forward each data packet to the next-hop corresponding to the destination specified in the packet.

2.2 Separating control and data plane security

The goal of routing in a network is to enable any pair of nodes within the network to communicate with each other. The goal of secure routing is to achieve the same objective in the face of adversarial nodes that attempt to disrupt inter-node communication. In order to achieve secure routing, it is essential to secure both the control-plane and the data-plane of routing.

An adversary (or a set of adversaries) may perform a different set of actions at the control-plane and the data-plane to disrupt routing. In the control-plane, the primary objective of an adversary is to propagate incorrect routing messages and attempt to hijack routes to different destinations. Additionally, an adversary in the control plane can drop routing messages that traverse it or generate spurious routing messages on behalf of other nodes. In the data-plane, an adversarial node that is already present along the routing path to a destination can either drop packets or misroute packets *i.e.*, route packets to neighboring nodes that are not the next-hop nodes in the routing path corresponding to the packet destinations.

Securing the control plane and data plane are two different problems given that the adversarial behavior is very different in the two planes. In the control plane, the goal of secure routing is to prevent adversarial nodes from propagating routing updates to hijack routes to different destinations. Securing the data-plane involves determining whether the underlying routing paths setup in the control plane work in practice.

Control-plane security does not imply data-plane security since node misbehavior in the data-plane (*e.g.*, drop, misroute packets) cannot be detected by security mechanisms in the control-plane. Consider the case where a router along the data-plane path has a forwarding error where it does not

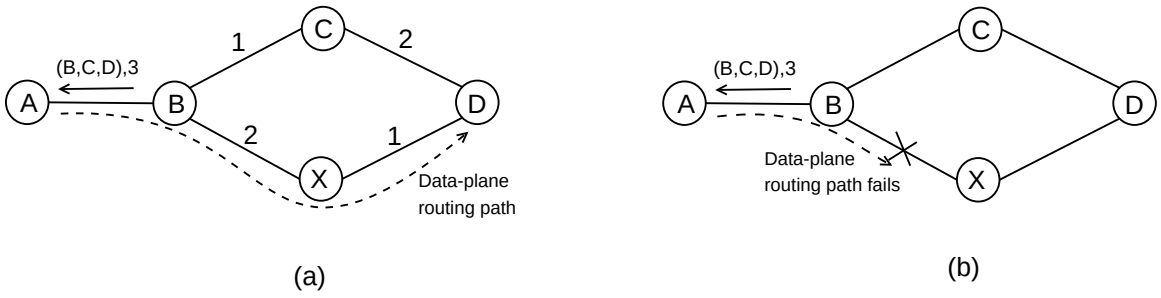


Figure 2.2: (a) Example of a path mismatch between the control plane and the data plane. (b) In the presence of a path mismatch, a failure in the data plane may not be visible to the control plane

forward data packets for a specific destination. In this case, from a control plane perspective, the route will appear to be perfectly functional while the underlying route in the data-plane is non-functional. In this case, verifying the correctness of routing announcements in the control plane is insufficient to address this data-plane problem. Forwarding errors are known to occur in practice [120, 81, 129]; in Chapter 3, we briefly describe some of the forwarding errors we detected in our experiments.

Another motivating reason to separate control-plane verification and data-plane verification is the case of *route mismatches* between the control plane and data plane *i.e.*, the underlying routing path in the data plane does not match the control plane. An example of a control and data-plane mismatch is illustrated in Figure 2.2(a) where the control plane path advertised by node B for destination D, namely (B, C, D), differs from the underlying routing path in the data-plane, namely (B, X, D). In the context of Internet routing, one possible reason for control-data plane mismatch is *stale routes* where a router artificially limits propagation of new routing advertisements during periods of huge spikes of simultaneous routing updates. In such a case, the route selection process can shift the route from (B, C, D) to (B, X, D) while the routing protocol does not propagate this update to other neighbors thereby making the previous routing announcement to become stale. The work

by Mao *et al*. [81] cites several reasons that can trigger a control-data plane mismatch in Internet routing.

Control-data plane route mismatches motivate the need for a separate mechanism for verifying the correctness of the data-plane. Consider the case where a link-failure occurs along the data-plane routing path to a destination where the control plane path is different from the data-plane path. In this case, if the control and data plane paths mismatch, the link-failure may not be visible to the control plane. For example, in Figure 2.2(b), if the link (B, X) fails, then the control plane does not observe the failure.

2.3 Secure routing problem definition

In this section, we will define two versions of the secure routing problem from the perspectives of both the control-plane and the data-plane.

2.3.1 Control plane security problem

Threat Model: We consider two basic forms of adversaries: *independent* and *colluding*. An *adversarial* node can perform two types of actions: (a) discard messages traversing the node; (b) generate spurious messages. Colluding adversaries have the additional capability of tunneling routing information between themselves during protocol execution while independent adversaries cannot. If there exists only one adversary in the system, such an adversary is referred to as an *isolated adversary*.

Let G denote the underlying network on n nodes. Let k represent the number of adversarial nodes

and the remaining $n - k$ are good nodes that obey a specified routing protocol. We define a *genuine route* to be one that only contains edges present in the network, G. Additionally, in any routing protocol, every node is initially aware of only its neighbors but is *unaware* of the rest of the network. We define the control plane secure routing problem as follows:

Control-plane secure routing problem: *Consider a graph G of n nodes where each node is initially aware of only its neighbors and every node is aware of a value k which represents a bound on the number of adversaries. Under what constraints can every good node discover genuine routes to every other good node in the network?*

This problem definition implicitly assumes that every node is aware of a bound on the number of adversaries, without which one cannot theoretically solve the secure routing problem. In Chapter 6, we describe how nodes can determine the appropriate value for k based on the topology.

One of the important aspects to note regarding this problem definition is *routing protocol independence*. The secure routing problem as we have defined it is independent of any routing protocol and captures the basic essence of what is required to achieve secure routing. Many of the secure routing problem definitions in prior work [100, 72, 65, 61, 91, 128, 38, 115, 62, 127] is protocol specific. For example, prior work on securing link-state routing [100, 91, 38] or distance-vector routing [127, 115, 62] focus on verifying the correctness of each routing message and hence, the corresponding techniques developed for securing these protocols are closely tied to the structure of the routing protocol. By formulating a protocol independent version of the secure routing problem, we are able to map it to the reliable communication problem, a classic problem in distributed systems which we will describe later in Section 2.4. The mapping to the reliable communication problem would be less straight-forward for a protocol dependent problem formulation. Due to the

protocol independent nature of the problem definition, we have been able to build a *reliable communication toolkit* which provides a generic set of security primitives that can be integrated across different types of routing protocols to achieve decentralized security. We will describe the toolkit in greater detail in Chapter 6.

There are two fundamental limitations to control-plane security. First, a genuine route can contain an adversarial node along the path provided the corresponding edges exist in G. If such an adversary drops data packets, detecting such attacks is a data-plane problem and outside the purview of control-plane security. Second, two colluding adversaries can always pretend the existence of a direct link by tunneling routing information/advertisements between them. In the absence of complete knowledge of the underlying topology, these fake links cannot be detected even using a public-key infrastructure. As mentioned earlier, packet drops on genuine routes using these fake links can be detected only in the data-plane.

2.3.2 Data-plane security problem

Threat model: An adversary along the routing path to a destination can perform two basic actions on data packets: (a) drop packets; (b) misroute packets to nodes not present on the routing path. Also, an adversarial node can potentially use any additional mechanisms to defend against the security mechanisms we build.

Securing the control plane is a requirement in building a secure data-plane. Once an adversary has successful inserted itself along a routing path, the adversary can drop packets. Control plane security deals with preventing adversaries from hijacking routes and inserting themselves in the data-plane routing path. Hence, before we secure the data plane, we assume that we have some mechanism

that can secure the control-plane.

In addition, from purely a data-plane perspective (with no information from the control plane), an adversary that drops packets in the data-plane can appear to be equivalent to a link/node failure that drops all packets. Hence, the data-plane can at most detect the presence of a problem but cannot distinguish a failure from adversary-induced packet drops. Also, the data-plane mechanism cannot correct the solution without the help of the control plane which is responsible for determining an alternate route. While the control plane may be unable to detect adversary induced packet losses, it relies on the data plane mechanism to provide feedback on the correctness of data-plane routes.

Correctness at the data-plane can be defined in two ways: (a) Did the packet reach the intended destination? (b)Did the packet traverse the same route established in the control plane? Based on this, we can define two variants of the data plane security problem as follows:

Data-plane secure routing problem (Rigid version): *Consider a network G on n nodes and a routing protocol R that establishes a genuine route between every pair of nodes in the control plane. Under what constraints, can we build a data-plane mechanism that can determine all invalid routes where the data-plane routing path does not match the control-plane routing path established by R?*

Data-plane secure routing problem (Relaxed version): *Consider a network G on n nodes and a routing protocol R that establishes a genuine route between every pair of nodes in the control plane. Under what constraints, can we build a data-plane mechanism that can determine whether every data-packet is reliably delivered to its intended destination?*

The data plane security mechanism is at best a verification mechanism to test whether the underlying routes set up in the control plane work or not. If a route is discovered to not work in the data plane, it is the responsibility of the control-plane to discover an alternate functional route. While the relaxed version merely tests whether each data-packet reaches the corresponding destination, the rigid version provides a more complete specification where it verifies whether the control and data planes match. Additionally, if we restrict the role of an adversary in the data plane to a *passive* adversary that merely drops all data packets and does not generate responses to thwart our security mechanisms, then we can detect such incorrect routes traversing such an adversary using the simple *Listen* protocol we describe in Chapter 3. Forwarding errors in routers or stale routes are examples of passive adversarial nodes that induce data-plane problems that can be detected using Listen.

2.4 Relationship to reliable communication

Now, we will elaborate the relationship between the secure routing problem and reliable communication [45, 22, 23], a fundamental problem in distributed systems that arises in the context of the Byzantine agreement problem [76, 98]. The classic reliable communication problem can be stated as follows:

Reliable communication problem: *Consider a communication network G on n nodes and a bound k on the number of adversarial nodes. Every good node A needs to communicate a message $m(A, B)$ to every other node B in G. Under what constraints, can every pair of good nodes reliably communicate their messages?*

The secure routing problem naturally relates to the reliable communication problem. To achieve

control-plane security, we need to address two problems:

1. How does a good node reliably discover every other good node in the network?
2. How does a good node learn consistent routing information to the other nodes that it has discovered?

The first problem is equivalent to the reliable communication problem where every good node needs to reliably broadcast its identity to all the other nodes in the network. This provides the ability to establish a reliable channel between every pair of good nodes by requiring every node to reliably broadcast its public-key as part of its identity in the node discovery phase. Once the reliable communication channel is established, it can be used to address the second problem of reliable dissemination of routing information.

To secure the data-plane, again one would need reliable communication. In the relaxed version of the problem, the verifying node needs to reliably communication with the destination to obtain proof of reliable delivery of a packet. In the rigid version, the verifying node needs to establish reliable communication with every node along the path to the destination.

While there have been several prior results on the reliable communication problem, they do not directly translate to the secure routing problem. The original version of the reliable communication problem [76, 45, 98, 22] implicitly assumes that the entire network G is initially known to all the nodes or nodes use a prior key distribution mechanism to *sign* messages. Neither of these assumptions hold in the secure routing problem wherein each node is initially only aware of its neighbors and is not aware of the entire topology. Additionally, there is no form of prior key distribution to enable node authentication.

In our work, we address a new variant of the reliable communication problem in *unknown networks*, where each node is only aware of its neighbors and is unaware of the entire network. We define an unknown network as follows:

Definition 1: An *unknown network* $U(n, G, N)$ comprises of n nodes connected by an undirected graph G where each node: (a) has a unique identity it cannot fake; (b) knows the identities of its neighbors in G; (c) knows a value $N \geq n$ which represents a bound on the size of the network.

Given this definition, the reliable communication problem in unknown networks can be defined as:

Reliable communication in unknown networks: *Consider an unknown network* $U(n, G, N)$. *Assume that* k *among the* n *nodes act in an* adversarial *manner and the remaining* $n-k$ *are* good *nodes that follow a prescribed algorithm. Under what constraints does there exist a distributed algorithm that enables every pair of good nodes in* G *to reliably communicate between themselves?*

2.4.1 Need for the connectivity constraint

The need for the connectivity constraint is best motivated by a simple example topology shown in Figure 2.3, Let x represent an adversarial node and let the nodes x and y form a 2-vertex cut, where they separate the rest of the nodes into two groups: group A and group B. In other words, any path from a node in group A to a node in group B has to traverse either x or y. In this example, x can prevent nodes in group A to reliably communicate with nodes in group B.

To illustrate this better, consider the case where x modifies every message that traversing it. Consider a node $u \in A$ and a node $v \in B$. Any message that u transmits to v has to either traverse x or y or both. If x modifies a message m to m' from u, then the recipient v may receive two distinct

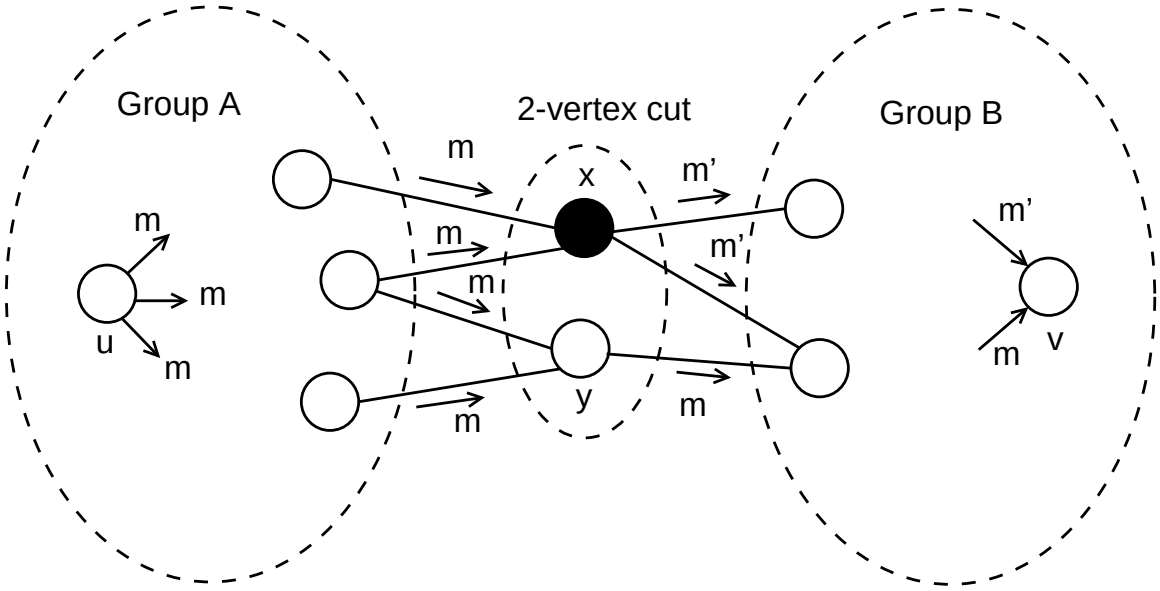


Figure 2.3: Example of a sparse network where a single adversary x can disrupt reliable communication between two groups of vertices A and B.

messages m and m' and cannot determine as to whether m or m' is the genuine message. Therefore u cannot reliably communicate with v. Here, in the absence of any prior secret exchange between u and v, neither of these nodes can determine as to whether x or y is adversarial.

This is an example of a 2-vertex connected graph (2 vertex disjoint paths exist between every pair of nodes) where a single adversary is sufficient to disrupt reliable communication. For a general graph, Dolev [45] proved that if there are k adversarial nodes, then every pair of nodes can reliably communicate if and only if the underlying graph is $2k + 1$ vertex connected. This result was established in the context of *known networks* where the entire graph is known in advance to all the nodes. We establish similar results in the context of *unknown networks* where each node is initially only aware of its neighbors but not the entire network.

2.4.2 Need for the fixed-identity criterion

Routing protocols operate in *unknown networks* where each node is initially aware of only its neighbors and not the rest of the network. In unknown networks, it is essential for the network to satisfy the *fixed-identity criterion* to be able to achieve decentralized security. The fixed identity criterion simply states that every node should have a unique identity that it does not fake to its neighbors. This assumption is essential to prevent an adversarial node from acting as multiple nodes using different identities. If the fixed-identity criterion is not met, then we show in Chapter 4 that a single adversarial node can disrupt reliable communication irrespective of the connectivity of the network. Prior results [37] have established the impossibility of establishing reliable communication in the face of Sybil attackers. The crux of the argument is that a Sybil attacker with N neighbors can simulate the behavior of N colluding adversaries.

The Internet and many social networks that we operate in today fall under the category of *unknown networks* that have nodes with fixed-identities. For example, the Internet topology comprises of roughly $17,000$ Autonomous systems (AS) where every AS has a unique identity (AS number) assigned by IANA [66]. When a new AS joins the network, it is only aware of the identities of its neighbors but is unaware of the AS topology. In fact, the complete AS graph structure of the Internet is unknown and is an open research problem to characterize the representativeness of the actual Internet topology collected from different measurement studies [34]. The Domain Name System (DNS) [88] and intra-domain routing are two other real-world examples of unknown networks with fixed-identities. Mobile ad-hoc networks [55] and P2P networks [94] are examples of two networks that do *not* fall into this category. The techniques we develop in this work are not applicable to networks which do not meet the fixed-identity criterion.

Enforcing the fixed identity constraint

Once a network has every node associated with a fixed-identity, additional work needs to be done at the protocol level to ensure that the fixed-identity criterion is adhered to *i.e.*, a node with a fixed identity does not modify it. In order to enforce the criterion, whenever a new node joins the network, there needs to be an explicit binding process associated with each neighbor that enforces the identity of the new node. Once the fixed-identity constraint is established between a node and its neighbors, we assume that the routing protocol adds basic checks to disallow an attacker that compromises the node from changing the node's identity. A link is broken if either party attempts to modify their identity. Hence, in our threat model, *misconfigurations and node compromises explicitly cannot break the fixed-identity criterion once it is established.*

2.5 Reliable communication: summary of key results

Figure 2.4 presents a complete summary of our theoretical results on the reliable communication problem in unknown networks. We will elaborate on these results in Chapters 4 and 5. The following results are important to note:

1. In the face of k adversaries, we require a minimum level of vertex connectivity to achieve reliable communication. A network that does meet this connectivity requirement is defined as a *dense network*; a network that does not meet this requirement is defined as a *sparse network*.
2. In sparse networks, where the minimum vertex-connectivity constraint is not met, our goal is to limit the damage that adversarial nodes may cause. In a dense network, our goal is to achieve reliable communication.

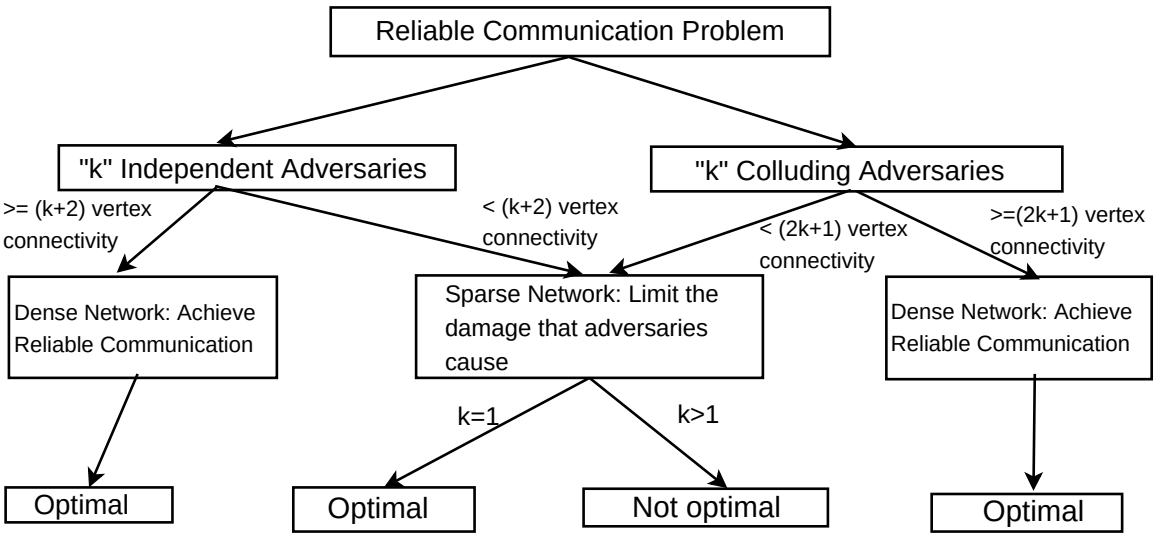


Figure 2.4: Summary of key results on reliable communication. In this figure, k represents the number of adversaries.

3. To handle k colluding adversaries, a minimum vertex-connectivity of $2k + 1$ is a necessary and sufficient constraint. However, to handle k independent adversaries, one requires only a minimum of $(k + 2)$ vertex connectivity.

4. In Chapter 5, we describe *penalty-based filtering* as a defense strategy to limit the damage of adversaries in sparse networks. We show that penalty-based filtering is the optimal defense strategy in the face of a single adversary. However for $k > 1$, the optimal strategy is still an open research question and we apply the same technique that we developed for the single adversary case.

2.6 Prior Work on secure routing

There is an enormous body of research literature in the space of secure routing. Most of these proposals to secure routing protocols leverage some form of public key distribution or prior key distribution. In this section, we will summarize the important works that are closely related to our work.

2.6.1 Byzantine robust routing

Perlman's thesis work [100] on Byzantine robustness of network protocols represents one of the early seminal works which describes mechanisms for securing network protocols in the presence of Byzantine failures *i.e.*, nodes through malice or malfunction exhibit arbitrary behavior such as corrupting, forging, dropping or delaying routing protocol messages. Specifically, her thesis provides a detailed description of how to secure link-state routing using the techniques developed in her thesis. All these mechanisms described in her work assume some form of prior key distribution using a "trusted server".

We have integrated some of these techniques in our reliable communication toolkit to secure link-state routing. In our setting, reliable communication provides decentralized key distribution assuming the underlying graph satisfies the necessary constraints outlined earlier in Section 2.4.1.

2.6.2 Secure Interdomain routing

There have several proposals in the past few years to secure BGP, the current inter-domain routing protocol. We classify these into two categories: (a) Key-distribution based approaches; (b) Non-PKI

approaches.

Key-distribution based approaches: Several different security mechanisms that use a PKI to sign routing updates have been proposed for BGP. These include the works by Smith *et al*. [114], Murphy *et al.* [92], *Secure BGP (S-BGP)* by Kent *et al.* [72, 70], Hu *et al* [65] and *Secure Origin BGP (soBGP)* [112]. All these protocols make extensive use of digital signatures and public key certification. A *digital signature* is simply a signature proof associated with any message that proves that the message indeed originated at the specified source (specified as part of the message). We will now briefly describe each of these proposals:

1. *Hop-by-hop authentication:* In BGP, routers use TCP connections to communicate routing messages. In hop-by-hop authentication, neighboring routers exchange symmetric public keys and sign every message using the key to prevent malicious adversaries from injecting invalid messages over the TCP connection by spoofing the source. The work by Smith and Garcia [114] identifies some of the weaknesses of BGP and propose basic counter measures using digital signatures to achieve hop-by-hop authentication. TCP-MD5 authentication, a more efficient authentication mechanism, has also been proposed for hop-by-hop authentication [59].

2. *Secure BGP (S-BGP):* S-BGP [72, 70] associates every routing update with a signature that protects all the fields within each routing update. S-BGP assumes a public-key infrastructure rooted at ICANN, the body responsible for address allocation, to distribute public keys for authentication. The Internet IP address space is divided into various address blocks called *prefixes* as allocated by the ICANN. Every BGP announcement corresponds to a specific prefix and the route corresponding to a prefix. In order to prevent unauthorized prefix adver-

tisements, S-BGP uses *address attestations* where the owner of a prefix signs a delegation message to enable the prefix to be advertised. Subsequently, every autonomous system (AS) along the path signs a *route attestation* for the path up to and including its identity. Each of these attestations can be verified using the PKI. S-BGP provides entire route validation while many other security proposals for BGP do not provide this guarantee.

3. *Secure Origin BGP (soBGP):* soBGP [112] is an effort to secure BGP where every router keeps an entire database of the network topology, BGP policy information as well as trusted certificates. The routers use this database to assess the authenticity of route updates. soBGP is specifically well tailored for origin authentication but is not well suited for validating an entire route. Additionally, soBGP cannot handle the case of modifications to the database cleanly; especially when routers or autonomous systems modify their public keys or if prefix ownership is modified.

4. *Secure Path-Vector routing (SPV):* SPV [65] is a recent proposal that proposes several cryptographic enhancements that make the public-key cryptography operations in signing BGP route updates much more efficient. One of the key ideas used in SPV to achieve an order of magnitude improvement is to use *Merkle Hash Trees* [83]. Merkle Hash Trees are an efficient signature mechanism that reduces the problem of authenticating a sequence of values to that of authenticating a single value. This reduces the number of signature verification operations that each router needs to perform. We refer the reader to [83] for a detailed explanation of Merkle hash trees.

Compared to the other proposals, S-BGP [72, 70] and soBGP [112] have gained a lot of traction having been refined and standardized by IETF working groups. However, none of these proposals

have moved towards adoption given the deployment hurdles associated with a Public Key Infrastructure [43, 49].

Non-PKI approaches: All the non-PKI approaches currently known for BGP do not provide any form of security guarantees in the face of deliberate attacks. However, some of these mechanisms are well-suited to handle configuration errors. Mahajan *et al.* [80] and Zhao *et al.* [133] provide mechanisms for detecting misconfigurations by correlating route advertisements in the control plane from several vantage points. One of the fundamental limitations of detecting misconfigurations based on analyzing BGP streams is *the lack of knowledge of the Internet topology*. Since the topology is not known, these techniques can pinpoint invalid routes only when the destination AS is wrongly specified but not when the path is modified. Other proposals for verifying the correctness of routing announcements include the *Internet Routing Registry* [5] and the *Inter-domain Route Validation Service* proposed by Goodell *et al.* [57]. These mechanisms assume the existence of databases with up to date authoritative route information which can aid routers to verify the route announcements that they receive. In these mechanisms, the routers contact authoritative route servers to verify the validity of routing information. However, the problem is to ascertain the authenticity, completeness, and availability of the information in such a database. First, ISPs are reluctant to submit routing information because this may disclose local policies that the ISPs regard as confidential. Second, the origin authentication of the database contents again demands a public key infrastructure [93]. Third, access to such databases relies on the very infrastructure that it is meant to protect, which is hardly an ideal situation.

2.6.3 Secure Distance Vector Routing

Hu *et al.* [62] propose several efficient mechanisms using one-way hash chains and authentication trees for securing distance-vector (DV) routing protocols. Their approach is one of the first attempts to authenticate the factual correctness of DV routing updates, and can prevent shorter and same distance fraud. It can also prevent newer sequence number fraud if a sequence number is used to indicate the freshness of a routing update. However, it does not address the case where an adversary increases the path length corresponding to a route. We use some of these constructions while integrating distance vector routing in our reliable communication toolkit as described in Chapter 6.

Routing Information Protocol (RIP) [36] is one of the first protocols that was adopted for intra-domain routing. RIP is a distance-vector routing protocol and is well-suited primarily for networks with a small number of nodes; however, RIP did not scale to large network sizes. Like most of the other routing protocols, RIP is also prone to the control plane security threat of incorrect routing announcements from adversarial nodes. S-RIP [127] is a secure distance vector routing protocol which uses a reputation based mechanism as a countermeasure to address the threats of router impersonation as well as to detect incorrect routing announcements involving an invalid path length. S-RIP only works in the presence of non-colluding adversaries. Pei *et al.* [99] propose a triangle theorem for detecting potentially or probably invalid RIP advertisements. Probing messages based on UDP and ICMP are used to further determine the validity of a questionable route. One disadvantage with this mechanism is that probing messages may be manipulated by adversarial nodes. A node advertising an invalid route can convince a receiver that route is valid by: a) manipulating the TTL value in a probing message; or b) sending back an ICMP message (port unreachable) on behalf of the destination. To prevent probe packet manipulation, one would either require all-pair reliable

communication to be established or have a prior key distribution mechanism to sign messages.

2.6.4 Secure Link-state routing

Open Shortest Path First (OSPF) [90], a link-state routing protocol, was later developed to address some of the limitations of RIP. The thesis work by Perlman [100] outlines the basic set of techniques essential to secure link-state routing. The work by Murphy and Badger [91] builds upon Perlman's work and shows how these techniques can be extended to secure OSPF using digital signatures where every routing message is associated with a digital signature from the source. Later, Cheung [38] showed how routers can use message authentication schemes for reducing the cost of link-state routing. All these works assume a Public Key Infrastructure (PKI) for signing messages. While an Internet-wide PKI is hard to deploy, deploying a PKI within a domain is relatively easier largely because the entire domain is under a single administrative entity.

2.6.5 Data-plane secure routing

All the prior related works that we mentioned focused on control-plane security. Data-plane security, in contrast, has received far less attention. In the context of BGP, Mao *et al.* [81] have build an AS-traceroute tool to detect the AS path in the data plane which can be used for data-plane verification. While this tool can detect several forms of invalid routes in the data plane, it is useful for diagnostic purposes only once a problem is detected. Padmanabhan *et al.* [95] propose a secure variant of `traceroute` to test the correctness of a route. However, this mechanism requires a prior distribution of cryptographic keys to the participating ASs to ascertain the integrity and authenticity of traceroute packets. In the design of feedback-based routing, Zhu *et al.* [134] proposed

a data plane technique based on passive and active probing. The passive probing aspect of this work shares some similarities to our Listen method which we will describe in Chapter 3.

Mizrak *et al.* [85] use a distributed collaborative mechanism between different routers to detect and isolate malicious routers with incorrect packet forwarding behavior in the network. They map the problem of detecting compromised routers as an instance of anomalous behavior-based intrusion detection where a compromised router is identified by correct routers when it deviates from exhibiting expected behavior. While this mechanism is well suited to detect routers with forwarding errors, it cannot handle the case of malicious adversaries that can potentially fool the learning based system.

2.7 Summary

In this chapter, we describe the secure routing problem, establish its correspondence to the reliable communication problem and discuss prior works on secure routing. The primary take-away messages from this chapter are four-fold:

1. Securing the control plane and the data plane of routing are two separate problems and one requires different mechanisms for addressing them.

2. The secure routing problem (both from the control plane and the data plane perspectives) is closely related to the reliable communication problem. However, the assumptions made by prior work on reliable communication do not satisfy the constraints of the secure routing problem. To achieve secure routing, we need to address a variant of the reliable communication problem for unknown networks where each node is initially only aware of its neighbors

and not the entire network.

3. A network should satisfy the *fixed-identity constraint* and the *connectivity constraint* to achieve reliable communication in an unknown network.

4. Finally, the problem of decentralized secure routing has received little attention in previous work. Previous work on secure routing primarily assume the existence of some form of prior key distribution for node authentication and signing routing messages.

In the next chapter, we consider the Border Gateway Protocol (BGP) as a specific case study and show how one can secure BGP in the face of adversaries.

Chapter 3

Securing the Border Gateway protocol

"We will bankrupt ourselves in the vain search for absolute security."

– Dwight David Eisenhower, American President

In this chapter, we consider the Border Gateway Protocol (BGP) as a specific case study and describe two decentralized security mechanisms, namely *Whisper* and *Listen* that we have developed for BGP. In the design of these security mechanisms, we abandon the goal for "perfect security" (as provided by a Public Key Infrastructure) and instead seek for "significantly improved security" through more easily deployable mechanisms. Both Whisper and Listen do not rely on any form of prior key distribution or on a central authority. Whisper is a control-plane verification mechanism that forms one of the basic building blocks for achieving reliable communication as illustrated later in Chapter 4 and Chapter 5. Listen, on the other hand, is a data-plane verification that verifies the validity of routes in the data-plane.

The chapter is organized as follows. In Section 3.1, we describe a brief overview of BGP and follow

that up with a brief introduction of the threat model we are dealing with in BGP in Section 3.2. Then, we summarize the security guarantees that our techniques in Section 3.3. Next, in Section 3.4, we describe the Whisper mechanism for control-plane verification and follow it up in Section 3.5 with a description of the Listen mechanism. In Section 3.6, we will describe our implementation of Listen and Whisper and describe its system overhead characteristics. We later perform a detailed performance evaluation of both these techniques using a real-world deployment in Section 3.7. Later in Section 3.8, we expose certain types of new attacks that colluding adversaries can launch by exploiting policy routing to cause more damage. Finally, we conclude with a summary of this work in Section 3.10.

3.1 Border Gateway Protocol: A brief introduction

The Internet is a network of over $17,000$ Autonomous Systems where each Autonomous System (AS) is by itself a network comprising of several routers under a single administrative entity. The Border Gateway Protocol (BGP) is the current de-facto routing protocol used in the Internet to maintain connectivity between ASs. Routers within an AS externally peer with routers in other ASs and exchange routing information using BGP.

For the technical material presented in this chapter, the following three basic aspects about BGP are important to note:

1. BGP performs *prefix-level routing*. A prefix represents an address block which is a portion of the Internet IP address space. Each institution or network is allocated ownership for a specific prefix by Internet Corporation for Assigned Names and Numbers (ICANN) [3]. BGP

routes at the granularity of prefixes and every entry in the BGP routing table corresponds to a specific prefix. To route a packet to a destination IP address, a BGP router performs a *longest prefix match* and routes the packet to the corresponding next hop.

2. BGP is a *path-vector routing* protocol at the AS level. Every AS is associated with a unique AS number which is allocated by the Internet Assigned Numbers Authority (IANA) [66]. Every BGP routing message is associated with a prefix and an *AS path* which contains the identities of all the ASs along the path to the destination network.

3. BGP uses *policy routing* where routers can use their local preferences for choosing routes to different destination. The policy choices of an AS need not follow any standard routing metric choice like the shortest path metric. Additionally, every AS can use its own *import* and *export* policy rules which dictate what routing advertisements from neighboring nodes are accepted (route advertisements can be ignored) and which advertisements are further propagated.

3.2 BGP threat model

The primary underlying vulnerability in BGP that we address in this work is the ability of an AS to create *invalid* routes. There are two types of invalid routes:

Invalid routes in the Control plane: This occurs when an AS propagates an advertisement with a fake AS path (i.e., one that does not exist in the Internet topology), causing other ASs to choose this route over genuine routes. A single malicious adversary can divert traffic to pass through it and then cause havoc by, for example, dropping packets (rendering destinations unreachable), eavesdropping (violating privacy), or impersonating end-hosts within the destination network (like Web servers

etc.).

Invalid routes in the Data Plane: This occurs when a router forwards packets in a manner inconsistent with the routing advertisements it has received or propagated; in short, the routing path in the data plane does not match the corresponding routing path advertised in the control plane. Mao *et al.* [81] show that for nearly 8% of Internet paths, the control plane and data plane paths do not match.

As described earlier in Chapter 2, invalid routes in the control plane can be triggered by configuration errors or malicious routers resulting from router compromises. Data plane, on the other hand, can be triggered by forwarding problems, malicious routers along the data path, stale routes (as illustrated in Chapter 2) and control-data plane route mismatches.

3.3 Summary:Listen and Whisper

In this work, we propose two decentralized mechanisms *Whisper* and *Listen* to secure the control and data plane of the Border Gateway Protocol. Whisper checks for consistency of routes in the control plane. Listen detects invalid routes in the data plane by checking whether data sent along routes reaches the intended destination. While both these techniques can be used in isolation, they are more useful when applied in conjunction. The extent to which they provide protection against the three threat scenarios can be summarized as follows:

Misconfigurations and Isolated Adversaries: Whisper guarantees *path integrity* for route advertisements in the presence of misconfigurations or isolated adversaries; *i.e.,* any invalid route advertisement due to a misconfiguration or isolated adversary with either a fake AS path or with any of the

fields of the AS path being tampered (*e.g.,* addition, modification or deletion of ASs) will be detected. Path integrity also implies that an isolated adversary cannot exploit BGP policies to create favorable invalid routes. In addition, Whisper can identify the offending router if it is propagating a significant number of invalid routes. Listen detects reachability problems caused by errors in the data plane, but is only applicable for destination prefixes that observe TCP traffic. However, none of our solutions can prevent malicious nodes already on the path to a particular destination from eavesdropping, impersonating, or dropping packets. In particular, countermeasures (from isolated adversaries already along the path) can defeat Listen's attempts to detect problems on the data path.

Colluding Adversaries: Two colluding nodes can always pretend the existence of a direct link between them by tunneling packets/ advertisements. In the absence of complete knowledge of the Internet topology, these fake links cannot be detected even using heavy-weight security solutions like Secure BGP [71]. While these fake links enable colluding adversaries to propagate invalid routes without being detected, we show that if BGP employs *shortest-path* routing then a large fraction of the paths with fake links can be avoided. On the contrary, colluding adversaries can exploit the current application of BGP policies to mount a large scale attack. To deal with this problem and yet support policy-based routing, we suggest simple modifications to the BGP policy engine which in combination with Whisper can largely restrict the damage that colluding adversaries can cause.

3.4 Whisper: Control Plane Verification

In this section, we will describe the whisper protocol, a control plane verification technique that proposes minor modifications to BGP to aid in detecting invalid routes from misconfigured or ma-

licious routers. In this section, we restrict our discussion to the case where an isolated adversary or a single misconfigured router propagates invalid routes. We will discuss colluding adversaries in Section 3.8.

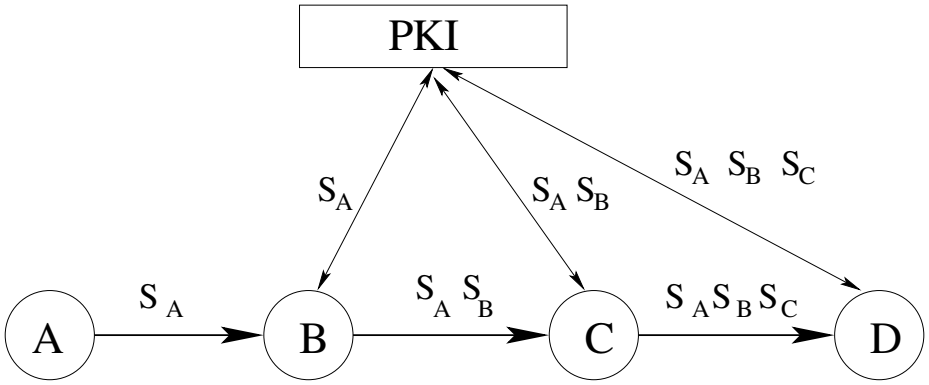
The Whisper protocol provides the following properties in the presence of isolated adversaries:

1. Any misconfigured or malicious router propagating an invalid route will always a trigger an alarm.
2. A single malicious router advertising more than a few invalid routes will be detected and the effects of these spurious routes will be contained.

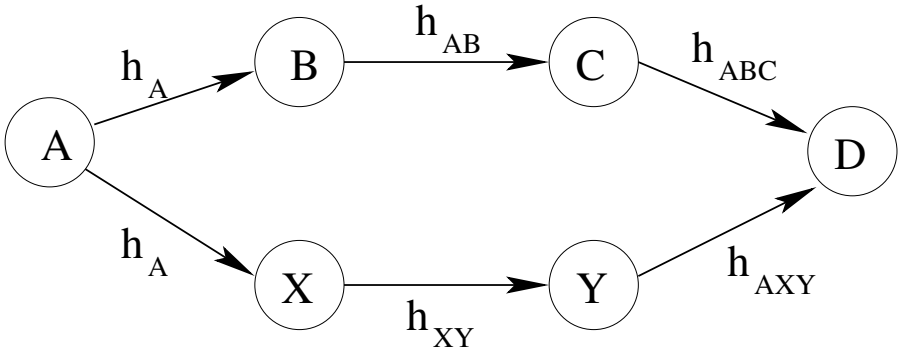
3.4.1 Triggering Alarms vs Identification

The main distinction between our approach and a PKI-based approach is the concept of *triggering alarms* as opposed to *identifying the source of problems*. In Secure-BGP, a router can verify the correctness of a single route advertisement by contacting a PKI and a central authority to test the validity of the signatures embedded in the advertisement . For example, in Figure 3.1 (Case(i)), each AS X appends an advertisement with a signature S_X generated using its public key. Another AS can use a PKI to check whether S_X is the correct signature of X. In this case, any misconfigured/malicious AS propagating an invalid route will not be able to append the correct signatures of other ASs and can be *identified*.

Without either of these two infra-structural pieces, a router cannot verify a single route advertisement in isolation. The Whisper model is to consider two different route advertisements to the same destination and check whether they are consistent with each other. For example, in Figure 3.1



Case(i): Secure−BGP model



Case(ii): Whisper Protocol Model

Figure 3.1: Comparison of the security approach of Whisper protocols with Secure BGP

Case(ii), each route advertisement is associated with a signature of an AS path. AS D receives two advertisements to destination A and can compare the signatures h_{ABC} and h_{AXY} to check whether the routes (C, B, A) and (Y, X, A) are consistent. When two routes are detected as *inconsistent*, the Whisper protocol can determine that at least one of the routes is invalid. However, it cannot clearly pinpoint the source of the invalid route. Upon detecting inconsistencies, the Whisper protocol can *trigger alarms* notifying operators about the existence of a problem. This method is based on the composition of well-known principles of *weak authentication* as discussed by Arkko and Nikander [20].

Whisper does not require the underlying Internet topology to have multiple disjoint paths to every destination AS. As long as an adversary propagating an invalid route is not on every path to the destination, whisper will have two routes to check for consistency: (a) the genuine route to the destination; (b) invalid path through the adversary.

3.4.2 Route Consistency Testing

A *route consistency test* takes two different route advertisements to the same destination as input and outputs *true* if the routes are consistent and outputs *false* otherwise. Consistency is abstractly defined as follows:

1. If both route announcements are valid then the output is *true*.
2. If one route announcement is valid and the other one is invalid then the output is *false*.
3. If both route announcements are invalid then the output is *true* or *false.*

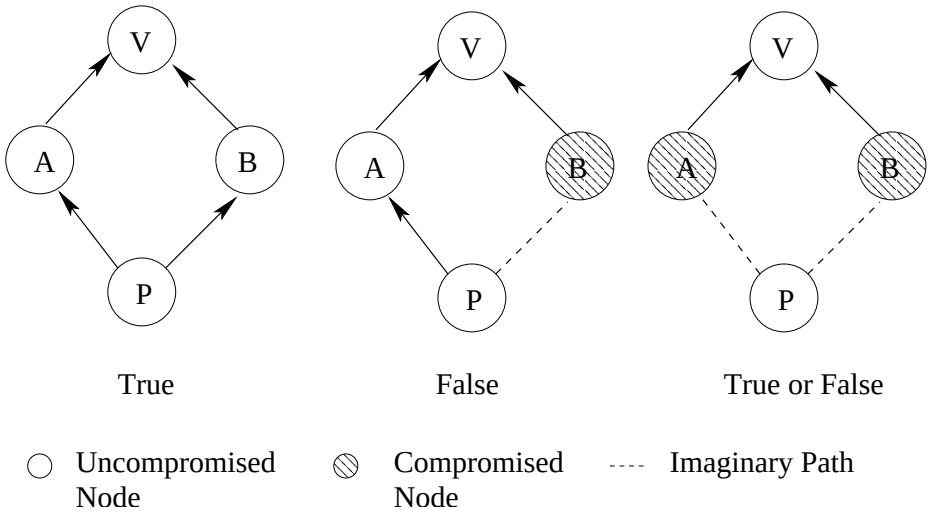


Figure 3.2: Different outcomes for a route consistency test. In all these scenarios, the verifying node is V. The verifying node checks whether the two routes it receives to destination P are consistent with each other.

The key output from a route consistency test is *false*. This output unambiguously signals that *at least one* of the two route announcements is invalid. In this case, our protocols can raise an alarm and flag both the suspicious routes as potential candidates for invalid routes. If the consistency test outputs true, both the routes could either be valid or invalid. Figure 3.2 depicts the outcomes of a route consistency test for various examples of network configurations.

We will now describe different flavors of route consistency tests of increasing complexity which offer different security guarantees. Conceptually, these constructions introduce a *signature* field in every BGP UPDATE message which is updated by every AS along a path and is used for performing the route consistency test. The origin AS (the originator of a route announcement) of a destination prefix initiates the signature field and every intermediary AS that is not the origin of a destination prefix is required to update the signature field using a cryptographic hash function.

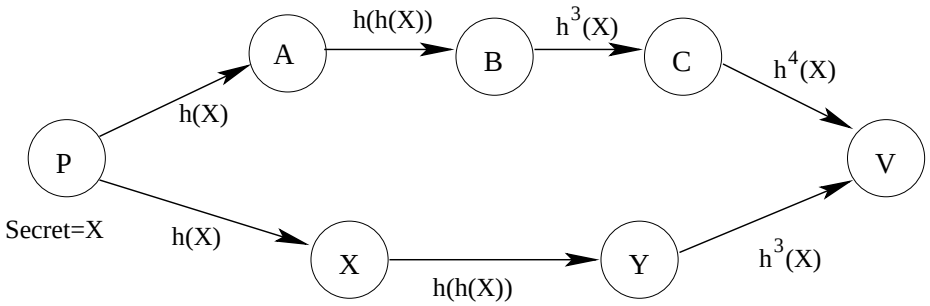


Figure 3.3: Weak-Split construction using a globally known hash function $h()$

Weak-Split Whisper(WSW): Figure 3.3 illustrates the weak-split construction using a simple example topology. Weak-Split whisper is motivated by the hash-chain construction used by Hu *et al.* [63, 62] in the context of ad-hoc networks. The key idea is as follows: The origin AS generates a secret x and propagates $h(x)$ to its neighbors where $h()$ is a globally known one-way hash function. Every intermediary AS in the path repeatedly hashes the signature field. An AS that receives two routes r and s of AS hop lengths k and l with signatures y_r and y_s can check for consistency by testing whether $h^{k-l}(y_s) = y_r$.

The security property that the weak-whisper guarantees is: *An independent adversary that is N AS hops away from an origin AS can propagate invalid routes of a minimum length of $N-1$ without being detected as inconsistent.* However, weak split whisper cannot offer path integrity since an adversary can modify the AS numbers along a path without affecting the path length.

The path integrity property requires the whisper protocol to satisfy two properties: (a) a malicious adversary should not be able to reverse engineer the signature field of an AS path; (b) any modification to the AS path or signature field in an advertisement should be detected as an *inconsistency* when tested with a valid route to the same destination.

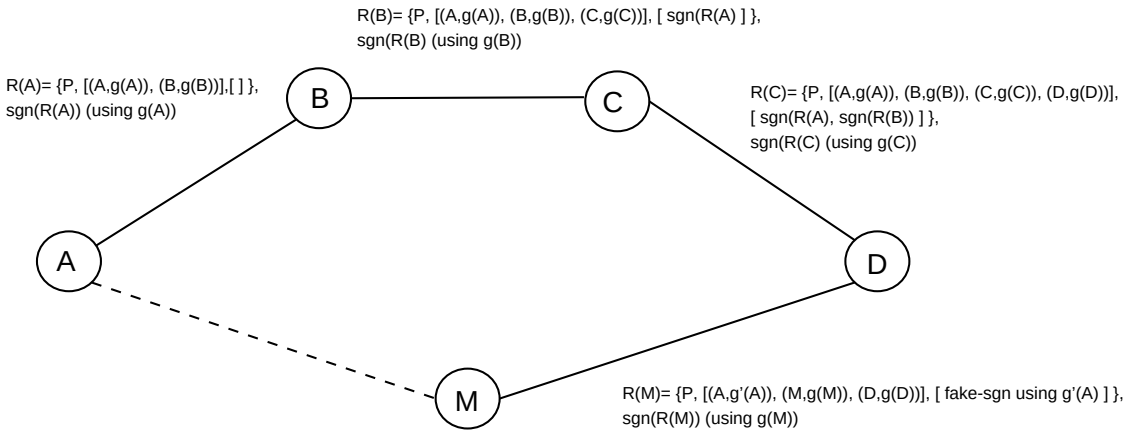


Figure 3.4: RSA-based Strong Split Whisper construction. When node M claims to have direct connectivity to A while in reality it does not, it has to generate a new public key for A, namely, $g'(A)$ (different from the actual public key $g(A)$) to generate a genuine signature. Hence, a public-key mismatch implies a route inconsistency.

RSA-based Strong Split Whisper

We motivate RSA-based Strong-Split Whisper (SSW) using an example topology illustrated in Figure 3.4 where every node represents a AS. In this topology, D has a genuine routing path (D, C, B, A) to node A and M is a malicious node that pretends to have a direct link to A. D receives two routing advertisements to A, one from C and one from M and needs a mechanism for determining that these two routing announcements are *inconsistent* with each other.

To achieve this property, we associate every routing announcement with a *signature* which is initialized by the source and incrementally updated by every node along the path. We use the example topology in Figure 3.4 to illustrate the signature construction. The signature construction has the following steps:

1. *Initialization:* In the initialization step (before propagating any route advertisement), every AS generates its own public key, private key pair and propagates its public key to its direct

neighbors. Hence A generates its own public-key $g(A)$ and claims its public key as $g(A)$ to B. Each node is associated with a *keyed-identity* which is the (identity, public-key) pair. The keyed-identity of A is $(A, g(A))$. However, an adversary can generate different public key claims for different neighbors but with respect to each neighbor it can have a single keyed identity.

2. *Originating a routing advertisement:* Let AS A in Figure 3.4 own prefix P. In order to advertise the prefix P to B, A generates the routing message $R_A = \{P, [(A, g(A)), (B, g(B))]\}$ that contains A and B's keyed identities. A generates a signature $sgn(R_A)$ where it signs R_A using its private key. A propagates $R_A, sgn(R_A)$ to B. Here, it is important to note that every AS appends the identity of its successor before propagating the route announcement to its successor. This prevents the successor from removing its identity from the routing announcement. In this case, B cannot remove its keyed identity claim without modifying the keyed identity of A. If B attempts to remove its identity, it has to generate a new public-key claim for A to generate a corresponding signature.

3. *Updating a routing advertisement:* Updating a routing advertisement occurs in the same fashion as the signature generation by the origin. Every routing advertisement is associated with a *keyed-identity path* and a *chain of signatures*. The keyed-identity path contains the entire AS path as well as the public-key claims of the all the ASs along the path. The chain of signatures is the corresponding set of signatures generated by every AS along the path. In Figure 3.4, when B receives the advertisement R_A along with its signature $sgn(R_A)$, then B constructs the routing message $R_B = \{P, [(A, g(A)), (B, g(B)), (C, g(C))], [sgn(R_A)]\}$. It then propagates R_B along with its signature $sgn(R_B)$ generated using its private key. There

are two aspects to note here. First, B includes $sgn(R_A)$ as part of R_B and signs it as well. This creates a linkage across signatures which makes it difficult for an adversary to modify and replace individual pieces of the signature. Second, when C receives both $sgn(R_A)$ and $sgn(R_B)$ in the routing announcement.

4. *Verifying the signature:* Verifying the signature is a simple operation which applies well-known public-key cryptographic mechanisms. Given a message m, an RSA public key P and a signature $s(m)$, any node can verify whether $s(m)$ is signature for message m using the public-key P using well-known RSA mechanisms described in [108]. In this case, every routing advertisement contains a chain of signatures as well as the set of claimed public keys of the ASs along the path. Before propagating any routing announcement, a node needs to verify whether each signature generated by an AS along the path indeed corresponds with the claimed public key in the announcement. For example, C needs to verify whether $sgn(R_A)$ and $sgn(R_B)$ are the signatures corresponding to the public key claims $g(A)$ and $g(B)$.

Route consistency testing: The above signature mechanism lays the foundation for testing whether two routing advertisements corresponding to a destination are *consistent* with each other. Consider the example topology in Figure 3.4 where M is an adversarial node that generates an incorrect routing announcement claiming direct connectivity to A while in reality it does not have a direct connection. In order to generate a corresponding signature that matches this announcement, M has to generate a new public-private key pair for A and sign the message using this new key pair. Hence, the keyed-identity corresponding to A (as generated by M) is $(A, g'(A)$ which is different from the actual keyed-identity $(A, g(A))$. Hence, when D receives two routing announcements to A (one from C and the other from M), D can determine that these announcements are inconsistent

with each other because the keyed-identity claims for A do not match. However, just based on these two routing announcements D cannot determine the genuine keyed identity corresponding to a destination AS. In Chapter 4, we present the reliable communication algorithm which determines the constraints under which D can determine the genuine keyed-identity corresponding to other ASs in the network. In summary, *two routing announcements to a destination are consistent with each other if the public-key claims corresponding to the two announcements match.*

The signature mechanism described above forms the basis of the reliable communication theory described in Chapters 4 and 5. The above construction leverages public-key cryptography which is relatively computationally expensive. We will show in Section 3.6.3, that the computational overhead imposed on routers by this technique is marginal.

We will now describe two alternate whisper constructions which offer path-integrity but do not involve public-key cryptography. However, these mechanisms would require significant modifications to be integrated into BGP. On the other hand, the RSA based SSW signature mechanism can be easily integrated into BGP today as shown in Section 3.6.1.

SHA-based Strong Split Whisper

The SHA-based Strong split whisper is an emulation of the previous RSA construction where the RSA signature operation is replaced with the SHA one-way hash function [110]. Unlike RSA-based SSW, SHA-based whisper signatures can only be verified by the originator.

Let $h_S()$ represent the SHA one-way hash-function which takes an arbitrary string as input and outputs a 160-bit hash value. A SHA-SSW signature of a route R consists of two parameters: (a) the

hash-value of the path; (b) public-key of the originator (as published in the route announcement). In SHA-based SSW, an origin A initiates an announcement to its neighbor B with the signature $h_S(Z,(A,B))$ where Z is a 160-bit nonce and P, its public key. Every intermediary AS B along a path that receives an update from a neighbor A with a SHA signature Y generates the signature $h_S(Y,(A,B,C))$ to its successor C along the path. Hence, a SHA whisper signature is simply a hash signature of the initiator's nonce Z and all neighbor bindings (A,B,C) along a path. The path integrity of SHA signatures follows because the SHA signatures are not: (a) invertible given the one-way hash function property; (b) reproducible by an adversary.

Consistency Testing: Unlike RSA, only the origin A can verify the correctness of the SHA signature of a path. A node V that receives two routes R,S to origin A performs the following operations for consistency testing. First, if the public keys advertised in routes R and S are inconsistent, then the routes are obviously inconsistent. Second, if the public-keys are the same, V chooses R as its routing path (by fixing its routing table) and sends the encrypted form of S's SHA signature to A querying whether the signature matches the path. A is supposed to send its response to V and signs it with its private key so that V can verify whether A indeed generated the message. Similarly, by setting S as the chosen route, A can verify R's signature. If A responds positively, the routes are deemed consistent. Note that the above test does not make any assumption about the nature of the path from A to V (*i.e.,* symmetric routing is not necessary) since A signs its response using its private key. However, it assumes that at least one valid reverse path exists from A to V. In summary, SHA-based SSW guarantees path-integrity but has the additional complexity of a pair of message exchanges between the verifier and the originator. From an implementation perspective, these messages are routed using normal IP routes and the only modification necessary

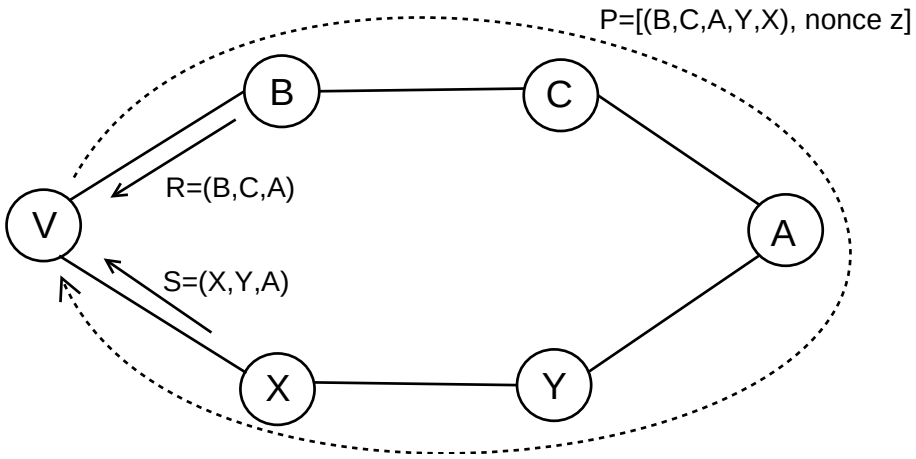


Figure 3.5: Loop Whisper: Source route a packet along a loop and test whether the packet is successfully received.

is an additional signature field in the BGP UPDATE message. We leverage two optimizations to reduce message overhead: (a) The public key of an origin needs to be communicated only once provided future updates use the same consistent public key. (b) Given that the set of distinct routes to a destination AS is relatively stable over time as well as small [33], the SHA signature verification needs to be done only once for each distinct AS path.

Loop Whisper

Loop Whisper is a simple consistency testing strategy which uses AS-level traceroute to check correctness. A verifier V that receives two route advertisements R and S to the same destination A can form an AS-loop involving itself and ASs in R and S. If R and S are completely vertex-disjoint (except the origin A), then the AS-loop is simply $R^{-1}S$ where R^{-1} is the inverse AS-path of R.

The loop whisper mechanism is illustrated in Figure 3.5. Given an AS-loop, the verifier generates a special control message (like an ICMP message) with a nonce and the AS-loop and *source-routes*

the message along the loop to test whether the loop exists (nonce is used as a packet identifier). Routing such control messages requires: (a) Each AS should have an additional control mechanism in the routers to handle these specific packets and route them to the neighbor as specified in the source route. (b) Each AS should forward control messages to a neighbor only when a genuine neighbor exists. The second constraint guarantees that if an adversary generates an invalid route with a non-existent path, the loop-test will never succeed. If a loop-test succeeds, two routes are deemed consistent. In summary, while loop whisper guarantees path integrity, it requires at least one router in each AS to support AS-level traceroute.(Note that not all routers need to be modified). From a deployment perspective, SHA-SSW signature based mechanism is easier to deploy than loop whisper.

3.4.3 Containment: Penalty Based Route Selection

Route consistency testing only provides the ability to trigger alarms whenever a node propagates invalid route announcements. We append consistency testing with *penalty based route selection*, a simple containment strategy that attempts to identify suspicious candidates and avoid routes propagated by them. The strategy works as follows: A router counts across destinations how often an AS appears on an invalid route, and assigns this count as a *penalty* value for the AS. The more destinations an adversary affects the higher becomes its penalty and the clearer it stands out from the rest. The route selection strategy is to *choose the route to a destination with the lowest penalty value.*

Consider the topology in Figure 3.6, where M is a malicious node that propagates 3 invalid route announcements with AS paths MA, MB, MC. By choosing the minimum penalty route, the

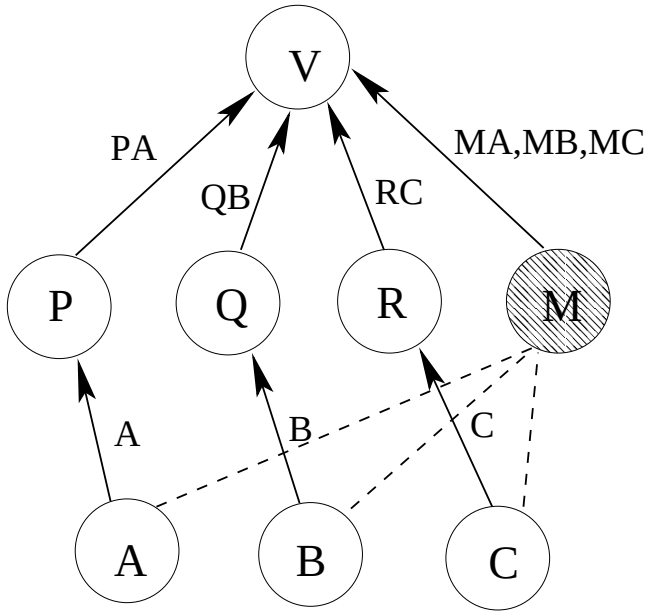


Figure 3.6: Detecting Suspicious ASs: In this example, M is a malicious AS that propagates 3 invalid routes to 3 different destinations A,B,C. The AS paths in the routes propagated are indicated along the links. The verifier V assigns penalty values of 3,$1, 1, 1$ to M, A, B, C respectively.

verifier V can avoid the invalid routes through M since they have a higher penalty value. One key assumption used in this technique is: *The identity of an AS propagating invalid routes is always present in the AS path attribute of the routes.* The identity of every AS is verified by the neighboring AS which receives the advertisement. For example, Zebra's BGP implementation [67] explicitly checks for this constraint for every announcement it receives. BGP should use shared keys across peering links to avoid man in the middle attacks.

Penalties should primarily be viewed as a reasonable first response to detect suspicious candidates and not as a fool-proof mechanism. In the presence of an isolated adversary, penalty based filtering can ensure that the effects of the adversary are contained. We believe that penalties is a good mechanism to detect malicious adversaries in customer ASs but should be applied with caution when involving ASs in the Internet core. In particular, penalties are not a good security measure

in the presence of colluding adversaries or when the number of independent adversaries is large. For example, multiple adversaries can artificially raise the penalty of an innocent AS by including its AS number in the invalid route. In Chapter 5, we show that a simple modified version of this penalty-based filtering strategy is the *optimal* defense strategy in the presence of a single adversary.

3.5 Listen: Data Plane Verification

In this section, we will present the Listen protocol, a data plane verification technique that detects reachability problems in the data plane. Reachability problems can occur due to a variety of reasons ranging from routing problems to misconfigurations to link failures. Listen primarily signals the existence of such problems as opposed to identifying the source or type of a problem.

Data plane verification mechanisms are necessary in two contexts: (a) connectivity problems due to stale routes or forwarding problems are detectable only by data plane solutions like Listen. (b) Blackhole attacks by malicious adversaries already present along a path to a destination. However, proactive malicious nodes can defeat any data plane solution by impersonating the behavior of a genuine end-hosts. The attractive features of Listen are: (a) passive; (b) incrementally deployable and standalone solution with no modifications to BGP; (c) quick detection of reachability problems for popular prefixes; (d) low overhead.

The basic form of the protocol described in this section is vulnerable to port scanners generating many incomplete connections. In Section 3.7.2, we use propose defensive measures against port scanners and motivate them using real world measurements.

procedure **LISTEN(P,T,N)**

Require: Prefix P, time period T, number of unique destinations N

```
1: t_0 = time at which first SYN packet observed
2: wait until |flows with distinct dest. in P| ≥ N
3: wait till clock time > t_0 + T
4: {Clean the data-set}
5: For every pair of IP addresses (src, dst) observed
6: if at least a single connection has completed then
7:    Add sample (src, dst, complete)
8: else
9:    Add sample (src, dst, incomplete)
10: end if
11: {Constants C_h, C_l must be determined in practice}
12: if fraction of complete connections > C_h then
13:    return "route is verifiable"
14: end if
15: if at least one connection completes then
16:    if fraction of complete connections < C_l then
17:       {Test for false positive}
18:       sample 2 future complete TCP flows towards P
19:       apply active dropping and retransmission checks
20:       if test is successful then
21:          return "route is verifiable"
22:       else
23:          return "route is not verifiable"
24:       end if
25:    end if
26: end if
```

Figure 3.7: Pseudo-code for the probing algorithm.

3.5.1 Listening to TCP flows

The general idea of Listen is to monitor TCP flows, and to draw conclusions about the state of a route from this information. The forward and reverse routing paths between two end-hosts can be different. Thus we may observe packets that flow in only one direction. We say that a TCP flow is *complete* if we observe a SYN packet followed by a DATA packet, and we say that it is *incomplete* if we observe only a SYN packet and no DATA packet over a period of 2 minutes (which is longer than the SYN timeout period).

Consider that a router receives a route announcement for a prefix P and wishes to verify whether prefix P is reachable via the advertised route. In the simplest case, a router concludes that the prefix P is reachable if it observes at least one complete TCP flow. On the other hand, the router cannot blindly conclude that a route is unreachable if it does not observe any complete connection. Incomplete connections can arise due to reasons other than just reachability problems. These include: (a) non-live destination hosts; (b) route changes during the connection setup of a single flow i.e. SYN and DATA packets traverse different routes. (c) port scanners generating SYN packets.

Under the assumption that port scanners are not present, detecting reachability problems would be easy. To deal with non-live destinations, a router should notice multiple incomplete connections to N different distinct destination addresses (for a reasonable choice of N). The problem of route changes can be avoided by observing flows over a minimum time period T. Hence, a router can conclude that a prefix is unreachable if during a period t it does not observe a complete TCP flow where t is defined as the *maximum* between: (a) the time taken to observe N or more incomplete TCP flows with different destinations within prefix P; (b) a predefined time period T.

The basic probing mechanism described above suffers from two forms of classification errors: (a) false negatives; (b) false positives. A false negative arises when a router infers a reachable prefix as being unreachable due to incomplete connections. A false positive arises when an unreachable prefix is inferred as being reachable. A malicious end-host can create false positives by generating bogus TCP connections with SYN and DATA packets without receiving ACKs. In Section 3.7.2, we show how to choose the parameters N and T to reduce the chances of incomplete connections causing false negatives.

Dealing with False Positives

Malicious end-hosts can create false positives by opening bogus TCP connections to keep a router from detecting that a particular route is stale or invalid. Adversaries noticing route advertisements from multiple vantage points (*e.g.,* Routeviews [125]) can potentially notice mis-configurations before routers notice reachability problems. Such adversaries can exploit the situation and open bogus TCP connections.

We propose a combination of *active dropping* and *retransmission checks* as a countermeasure to reduce the probability of false positives.

1. *Active dropping:* Choose a random subset of m_1 packets within a completed connection (or across connections), drop them and raise an alarm if these packets are *not* retransmitted. Alternatively, one can just delay packets at the router instead of dropping them.
2. *Retransmission check:* Sample a different random subset of m_2 packets and raise an alarm if more than 50% of the packets are retransmitted.

An adversary generating a bogus connection cannot decide which packets to retransmit without receiving ACKs. If the adversary blindly retransmits many packets to prevent being detected by Active dropping, the Retransmission check notices a problem. We set a threshold of 50% for retransmission checks assuming that *most* genuine TCP connections will not experience a loss-rate close to 50%.

Consider an adversary that has transmitted k packets in a TCP connection without receiving ACKs to retransmit a fraction, q, of these packets. Let $C(x, y) = \frac{x!}{(x-y)!.y!}$ represent the binomial coefficient for two values x and y. The probability with which the adversary is able to mislead the active dropping test is given by $\frac{C(k \cdot q, m_1)}{C(k, m_1)}$. The probability with which the retransmission check cannot detect an adversary is given by the tail of the binomial distribution $(1 - (\sum_{l=m_2/2}^{m_2} C(m_2, l) q^l (1 - q)^{m_2 - l}))$. Hence the overall probability, p_e, that our algorithm does not detect an adversary is:

$$\frac{C(k \cdot q, m_1)}{C(k, m_1)} \times (1 - (\sum_{l=m_2/2}^{m_2} C(m_2, l) q^l (1 - q)^{m_2 - l}))$$

For a given prefix, the overhead of active dropping can be made very small. By choosing $m_1 = 6$ and dropping only 6 packets across different TCP flows, we can reduce the probability of false positive, p_e, to be less than 0.1%.

This countermeasure is applied only when we notice a discrepancy across different TCP connections to the same destination prefix, *i.e.*, number of incomplete connections and complete connections are roughly the same. In this case, we sample and test whether a few complete connections are indeed bogus.

Detailed Algorithm

Figure 3.7 presents the pseudo-code for the listen algorithm. The algorithm takes a conservative approach towards determining whether a route is verifiable. Since false positive tests can impact the performance of a few flows, the algorithm uses the constant C_h and C_l to trade off between when to test for false positives. When the test is not applied, we use the fraction of complete connections as the only metric to determine whether the route works. The setting of C_h, C_l depends on the popularity of the prefixes. Firstly, we apply the false positive tests only for popular prefixes *i.e.,* $C_l = 0$ for non-popular prefixes. For a popular prefix, we choose a conservative estimate of C_h (closer to 1) *i.e.,* a large fraction of the connections have to complete in order to conclude that the route is verifiable. On the other hand, if we observe that a reasonable fraction of combination of incomplete connections, we apply the false positive test to 2 sampled complete connections. The user has choice in tuning C_l based on the total number of false positive tests that need to be performed. For non-popular prefixes, the statistical sample of connections is small. For such prefixes, we set the value of C_h to be small.

3.6 Implementation

In this section, we will describe the implementation of Listen and Whisper and their overhead characteristics.

3.6.1 Whisper Implementation

In this section, we will only focus on the implementation of the strong split whisper protocol (RSA variant). The SHA variant requires a modification to the hash function we use in our code. [1] The whisper implementation contains two basic components: (a) a stand alone whisper library which performs the cryptographic operations used in the protocol. (b) a Whisper-BGP interface which integrates the whisper functions into a BGP implementation. We implemented the Whisper library on top of the *crypto* library supported by OpenSSL development version 0.9.6b-33. We integrated this library with the Zebra BGP router implementation version 0.93b [67]. Our Whisper implementation works on Linux and FreeBSD platforms.

Whisper Library

The structure of a basic Whisper signature is:

```
typedef struct {
  BIGNUM *seed;
  BIGNUM *N;
}Signature;
```

BIGNUM is a basic data structure used within the OpenSSL crypto library to represent large numbers. The whisper library supports these three functions using the Signature data structure:

1: generate_signature(Signature *sg);
2: update_signature(Signature *sg, int asnumber, int position);
3: verify_signatures(Signature *r, Signature *s,int *aspath_r, int *aspath_s);

These functions exactly map to the three whisper operations described earlier in Section 3.4.2. The main advantage of separating the whisper library from the whisper-BGP interface is modularity. The

[1] The additional control messages in SHA-based SSW are data-plane messages and are not incorporated in the code.

whisper library can be used in isolation with any other BGP implementation sufficiently different from the Zebra version.

Integration with BGP

The Whisper protocol can be integrated with BGP without changing the basic packet format of BGP. BGP uses $32 - bit$ community attributes which are options within UPDATE messages that can be leveraged for embedding the signature attributes. This design offers us many advantages over updating a version of BGP. First, a single update message can have several community attributes and one can split a signature among multiple community attributes. Second, a community attribute can be set using the BGP configuration script to allow operators the flexibility to insert their own community attribute values. In a similar vein, one can imagine a stand-alone whisper library computing the signatures and a simple interface to insert these signatures within the community attributes. Third, one can reserve a portion of the community attribute space for whisper signatures. In today's BGP, community values can be set to any value as long as they are interpreted correctly by other routers. An RSA-SSW uses 2048 bits per signature field generated by each AS, while SHA-SSW needs a total of $1184 = 160 + 1024$ bits for the SHA signature and public key.

3.6.2 Listen Implementation

We implemented the passive probing component of *Listen* (i.e. without active dropping) in about 2000 lines of code in C and have ported the code to Linux and FreeBSD operating systems. The current prototype uses the *libpcap* utility [7] to capture all the packets off the network. This form of implementation has two advantages: (a) is stand-alone and can be implemented on any machine

(need not be a router) which can sniff network traffic; (b) does not require any support from router vendors. Additionally, one can execute *bgpd* (Zebra's BGP daemon [67]) to receive live BGP updates from a network router. For faster line-rates (e.g. links in ISPs), *listen* should be integrated with hardware or packet probing software like Cisco's Netflow [40]. The current implementation cannot support false positive tests since the code can only passively observe the traffic but cannot actively drop packets (since this does not perform the routing functionality).

In our implementation, the complexity of listening to a TCP flow is of the same order as a route lookup operation. Additionally, the state requirement is $O(1)$ for every prefix. We maintain a small hash table for every prefix entry corresponding to the (src,dst) IP addresses of a TCP flow and a time stamp. While a SYN packet sets a bit in the hash table, the DATA packet clears the bit and record a complete connection for the prefix. Using a small hash table, we can crudely estimate the number of complete and incomplete connections within a time-period T. Additionally, we sample flows to reduce the possibility of hash conflicts. This implementation uses simple statistical counter estimation techniques used to efficiently maintain statistics in routers. Hence, the basic form of Listen can be efficiently implemented in the fast path of today's routers.

Deployment: We deployed our *Listen* prototype to sniff on TCP traffic to and from a $/24$ prefix within our university. Additionally, we received BGP updates from the university campus router and constructed the list of prefixes in the routing table used by the edge router. The tool only needs to know the list of prefixes in the routing table and assumes a virtual route for every prefix. The Listen tool can report the list of verifiable and non-verifiable prefixes in real time. Additionally, the *Listen* algorithm is applied only by observing traffic in one direction (either outbound or inbound).

Operation	512-bit	1024-bit	2048-bit
update_signature	0.18 msec	0.45 msec	1.42 msec
verify_signatures	0.25 msec	0.6 msec	1.94 msec
generate_signature	0.4 sec	8.0 sec	68 sec

Table 3.1: Processing overhead of the Whisper operations on a 1.5 Ghz Pentium IV with 512 MB RAM.

3.6.3 Overhead Characteristics

Overhead of Whisper: One of the important requirements of any cryptography based solution is low complexity. We performed benchmarks to determine the processing overhead of the Whisper operations. Table 3.1 summarizes the average time required to perform the whisper operations for 3 different key sizes: $512-$ bit, $1024-$bit and $2048-$bit. As the key size increases, the RSA-based operations offer better security. Security experts recommend a minimum size of 1024 bit keys for better long-term security.

We make two observations about the overhead characteristics. First, the processing overhead for all these key sizes are well within the limits of the maximum load observed at routers. For 2048 bit keys, a node can process more than $42,000$ route advertisements within 1 minute. In comparison, the maximum number of route advertisements observed at a Sprint router is 9300 updates every minute [13]. For 1024 bit keys, Whisper can update and verify over $100,000$ route advertisements per minute. Second, *generate_signature()* is an expensive operation and can consume more than 1 sec per operation. However, this operation is performed only once over many days.

Overhead of Listen: By analyzing route updates for over 17 days in Routeviews [125], we observed that 99% of the routes in a routing table are stable for at least 1 hour. Based on data from a tier-1 ISP, we find that a router typically observes a maximum of 20000 active prefixes over a period of 1 hour

i.e., only 20000 prefixes observe any traffic. If the probing mechanism uses a statistical sample of 10 flows per prefix, the overhead of probing at the router is negligible. Essentially, the router needs to process 200000 flows in 3600 sec which translates to monitoring under 60 flows every second (equivalent to $O(60)$ routing lookups). Even if the number of active prefixes scales by a factor of 10, current router implementations can easily implement the passive probing aspect of Listen.

Active dropping and retransmission checks are applied only in the IP slow path and are invoked only when a prefix observes a combination of both incomplete and complete connections. To minimize the additional overhead of these operations, we restrict these checks to a few prefixes.

3.7 Evaluation

In this section, we evaluate the key properties of Listen and Whisper. Our evaluation is targeted at answering specific questions about Listen and Whisper:

1. How much security can Whisper provide in the face of isolated adversaries?
2. How useful is Listen in the real world? In particular, can it detect reachability problems?
3. How does Listen react in the presence of port scanners? How does one adapt to such port scanners?

We answer question (1) in Section 3.7.1, questions (2),(3) in Section 3.7.2. Our evaluation methodology is two-fold: (a) empirically evaluate the security properties of Whisper; (b) use a real-world deployment to determine usefulness of Listen. To evaluate the security properties of Whisper, it is necessary to determine the effects of the worst-case scenario which is better quantified using an empirical evaluation.

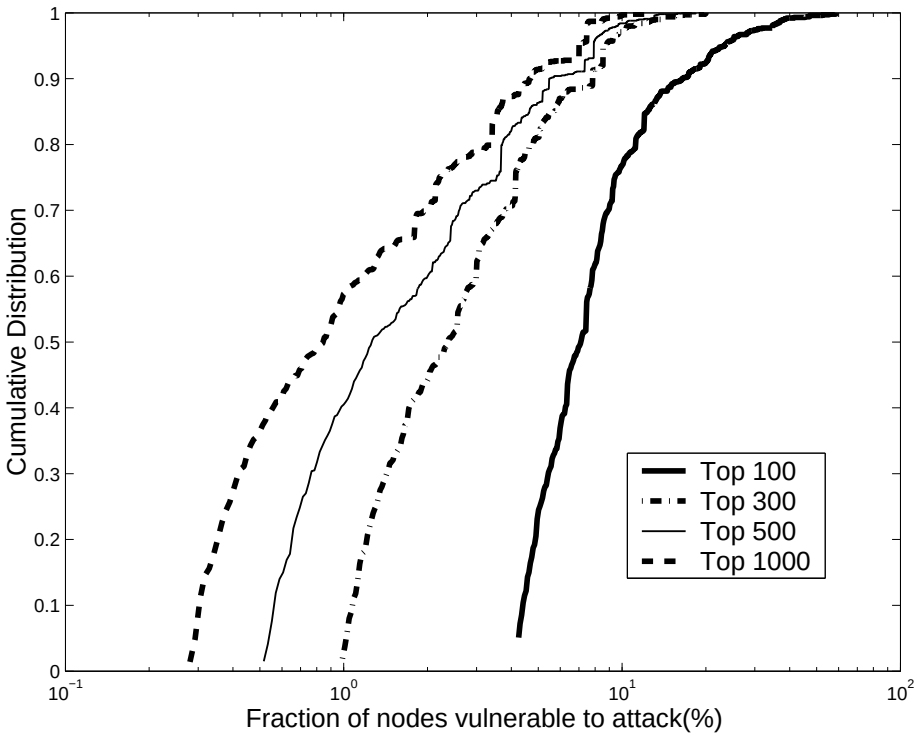


Figure 3.8: Effects of penalty based route selection

We collected the Internet AS topology data based on BGP advertisements observed from 15 different vantage points over 17 days including Routeviews [125] and RIPE [11]. The policy-based routing path between a pair of ASs is determined using customer–provider and peer–peer relationships, which have been inferred based on the technique used in [121].

3.7.1 Whisper: Security Properties against Isolated Adversaries

In this section, we quantify the maximum damage an isolated adversary can inflict on the Internet given that Strong Split Whisper is deployed. Since SHA-based SSW offers path integrity, an isolated adversary cannot propagate invalid routes without raising alarms unless there exists no alternate

route from the origin to the verifier (i.e. adversary is present in all paths from the origin to the Internet).

Given an adversary that is willing to raise alarms, we analyzed how many ASs can one such adversary affect. In this analysis, we exclude cases where the adversary is already present in the only routing path to a destination AS. We use penalty based route selection as the main defense to contain the effects of such invalid routes. We assume that in the worst-case, an adversary compromising a single router in an AS is equivalent to compromising the entire AS especially if all routers within the AS choose the invalid route propagated by the compromised router.

Let M represent an isolated adversary propagating an invalid route claiming direct connectivity to an origin AS O. AS V is said to be *affected* by the invalid route if V chooses the route through M rather than a genuine route to O either due to BGP policies or shorter hop length. Based on common practices, we associate all ASs with a simple policy where customer routes have the highest preference followed by peers and providers [54]. Given all these relationships, we define the *vulnerability* of an origin AS, O, as $V(O, M)$ to be the maximum fraction of ASs, M can affect. Given an isolated adversary M, we can quantify the worst-case effect that M can have on the Internet using the *cumulative distribution* of $V(O, M)$ across all origin ASs in the Internet.

With ASs deploying penalty based route selection as a defense, we expect the vulnerability $V(O, M)$ to reduce. We study how the cumulative distribution of $V(O, M)$ for a single adversary M varies as a function of how many ASs deploy penalty based route selection. We consider the scenario where the top n ISPs deploy penalty based route selection (based on AS degree). Figure 3.8 shows this cumulative distribution for for different values of $n = 100, 300, 500$ and 1000. These distributions are averaged across all possible choices for M.

We make the following observations. First, a median value of 1% for $n = 1000$ indicates that a randomly located adversary can affect at most 1% of destination ASs by propagating bogus advertisements assuming that the top 1000 ISPs use penalties. This is orders of magnitude better that what the current Internet can offer where a randomly located adversary can on an average affect nearly 30% of the routes (repeat the same analysis without SSW) to a randomly chosen destination AS.

Second, in the worst case, a single AS can at most affect 8% of the destination ASs for $n = 1000$. 8% is a limit imposed by the structure of the Internet topology since it represents the size of the largest connected without the top 1000 ISPs. One malicious AS in this component can potentially affect other ASs within the same component.

Third, if all provider ASs use penalties for route selection, the worst case behavior can be brought to a much smaller value than 8%. Additionally, there is very little benefit in deploying penalty based route selection in the end-host networks since they are not transit networks and typically are sources and sinks of route advertisements. Hence, any filtering at these end-hosts only protects themselves but not other ASs.

To summarize, the Whisper protocol in conjunction with penalty based route selection can guarantee that a randomly placed isolated adversary propagating invalid routes can affect at most 1% of the ASs in the Internet topology.

	Number of Reachability Problems	Probability of False Negatives
Outbound	235	0.93%
Inbound	343	0.37%

Table 3.2: Listen: Summary of Results

3.7.2 Listen: Experimental Evaluation

In this section, we describe our real-world experiences using the Listen protocol. We make two important observations from our analysis. First, we found that a large fraction of incomplete TCP connections are *spurious i.e.,* not indicative of a reachability problem. We show that by adaptively setting the parameters T, N of our listen algorithm we can drastically reduce the probability of such false negatives due to such connections. Second, we detect several reachability problems using Listen including specific misconfiguration related problems like forwarding errors. Table 3.2 presents a concise summary of the results obtained from our deployment. We detected reachability problems to 578 different prefixes with a very false negative probabilities of 0.95% and 0.37% respectively due to spurious outbound and inbound connections.

We will now describe our deployment experience in greater detail. In our testbed, we use three active probing tests to verify the correctness of results obtained using Listen: (a) ping the destination; (b) traceroute and check whether any IP address along in the path is in the same prefix as the destination; (c) perform a port 80 scan on the destination IP address. These tests are activated for every incomplete connection. We classify an incomplete connection as having a reachability problem only if all the three probing tests fail. We classify an incomplete connection as a *spurious connection* if one of the probing techniques is able to detect that the route to a destination prefix works. A spurious

Number of end-hosts behind $/24$ network	28
Number of days	40
Total No. of TCP connections	994234
No. of complete connections	894897
No. of incomplete connections	99337
Average Routing Table Size	123482
Total No. of Active Prefixes	11141
Average No. of Active Prefixes per hour	141
Average No. of Active Prefixes per day	2500-3000
Verifiable Prefixes	9711
Prefixes with perennial problems	42

Table 3.3: Aggregate characteristics of Listen from the deployment

TCP connection is an incomplete connection that is not indicative of a reachability problem.

Table 3.3 presents the aggregate characteristics of the traffic we observed from a $/24$ network for over 40 days. In reality, we found that nearly 10% of the connections are incomplete of which a large fraction of these connections are spurious (91% inbound and 63% outbound). A more careful observation at the spurious connections showed that nearly 90% of spurious inbound connections are due to port scanners and worms. The most prominent ones being the Microsoft NetBIOS worm and the SQL server worms [9]. Spurious outbound connections occur primarily due to failed connection attempts to non-live hosts and attempts to access a disabled ports of other end-hosts (*e.g.*, telnet port being disabled in a destination end-host).Given this alarmingly high number of spurious connections, we propose defensive measures to reduce the probability of false negatives due to such connections.

Defensive Measures to reduce False Negatives

In this section, we show that one can adaptively set the parameters N, T in the listen algorithm to drastically reduce the probability of false negatives due to spurious TCP connections. In particular, we show that by adaptively tuning the minimum time period, T, one can reduce false negatives due to port scanners and by tuning the number of distinct destinations, N, one can deal with non-live hosts.

Given the nature of incomplete connections in our testbed, we use outbound incomplete connections as a test sample for non-live hosts and inbound connections as the test sample for port scanners and worms. In both inbound and outbound, we restricted our samples to only those connections which are known to be false negatives.

Setting T: One possibility is to choose an interval T large enough such that the router will notice at least one genuine TCP flow during the interval. Such a value of T will depend on the popularity of a prefix. The popularity of a prefix, $pop(P)$, is defined as the mean time between two complete TCP connections to prefix P. We can model the arrival of TCP connections as a Poisson process with a mean arrival rate as $1/pop(P)$ [97]. Given this, we can set the value of $T = 4.6 \times pop(P)$ to be 99% certain that one would experience at least one genuine connection within the period T. To have a 99.9% certainty, one needs to set $T = 6.9 \times pop(P)$. For prefixes that hardly observe any traffic, the value of T will be very high implying that port scanners generating incomplete connections to such prefixes will not generate any false alarms.

From our testbed, we determine the mean separation time between the arrival of two incoming connections to be $pop(P) = 34.1$ sec. By merely setting $T = 156.8$ to achieve 99% certainty, we

Type of problem	Number of Prefixes
Routing Loops	51
Forwarding Errors	64
Generic (forward path)	146
Generic (reverse path)	317

Table 3.4: The number of prefixes affected by different types of reachability problems.

could reduce the probability of false negatives in Listen from 91.83% to 0.37%. Throughout the entire period of measurement, only during 8 periods of 156 seconds each did we verify incorrectly that the local prefix is not reachable.

Setting N: The choice of an appropriate value of N trades off between minimizing the false negative ratio due to non-live hosts and the number of reachability problems detected. In our testbed, we noticed that by merely setting $N = 2$, we can significantly reduce the false negative ratio in outbound connections from 63% to less than 1%. However, Listen reported only 35 out of 663 potential prefixes to have routing problems. For several $/24$ prefixes, we observed TCP connections to only a single host and by setting $N = 2$, we tend to omit these cases. In practice, the value of N is dependent on the diversity of traffic to a destination prefix and the traffic concentration at a router. For many $/24$ prefixes, we need to set $N = 1$. For $/8$ and $/16$ prefixes, one can choose larger values of $N = 4$ or $N = 5$ provided the prefix observes diversity in the traffic.

Detected Reachability Problems

Among the reachability problems detected by Listen, two specific types of routing problems (as detected by active probing) include: *routing loops* and *forwarding errors* due to unknown IP addresses. We detected routing loops using traceroute and inferred forwarding errors using the routing

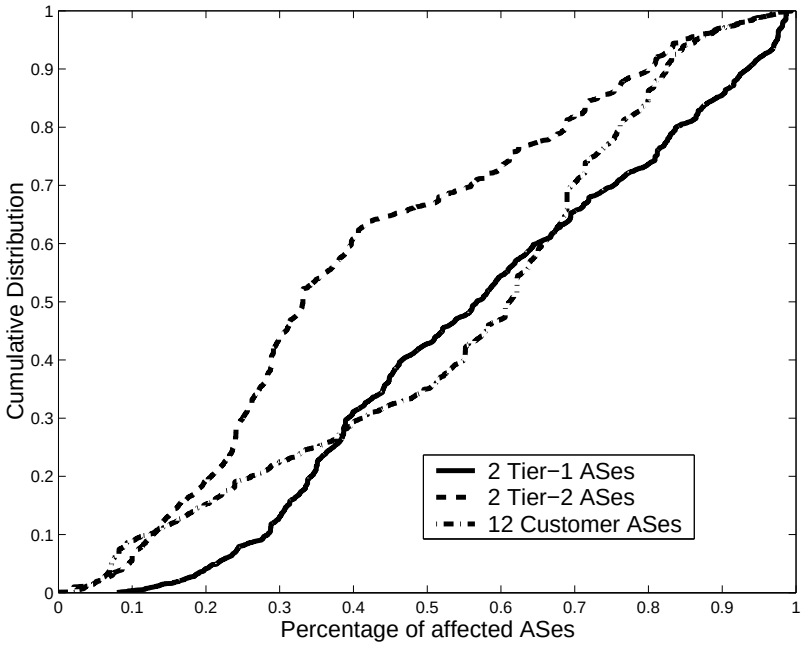


Figure 3.9: The effects of colluding adversaries in the current Internet.

table entries at the University exit router. A forwarding error arises when the destination IP address in a packet is a genuine one but the router has no next hop forwarding entry for the IP address. This can potentially arise due to staleness of routes. Table 3.4 summarizes the number of prefixes affected by each type of problem. In particular, we observe routing loops to 51 different prefixes and forwarding errors to 64 different prefixes. Additionally, Listen detected 463 prefixes having other forms of reachability problems.

To cite a few examples of reachability problems we observed: (a) A BGP daemon within our network attempted to connect to another such daemon within the destination prefix 193.148.15.0/24. The route to this prefix was perennially unreachable due to a routing loop. (b) The route to Yahoo-NET prefix 207.126.224.0/20 was fluctuating. During many periods, the route was detected as unavailable.

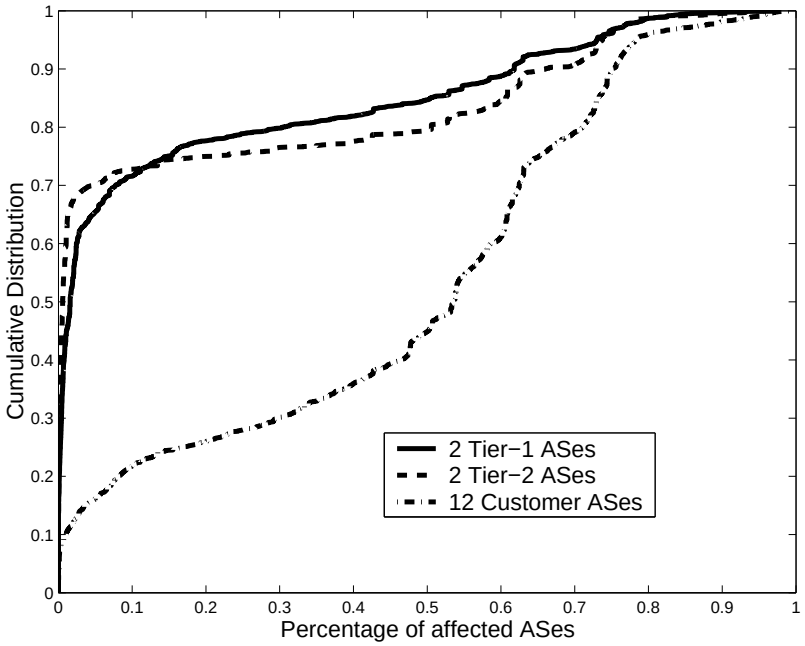


Figure 3.10: Effects of colluding adversaries with whisper + policy routing.

3.8 Colluding Adversaries

Additional to acting as a group of isolated adversaries, colluding adversaries can tunnel advertisements and secrets between them and create invalid routes with fake AS links without being detected by the Whisper protocols. These invalid routes are not detectable even with a PKI unless the complete topology is known and enforced. Despite the limitation, we can provide protective measures for avoiding these invalid routes.

Given the hierarchical nature and the skewed structure of the Internet topology, the invalid paths from colluding adversaries not detectable by the Whisper tend to be longer in AS path length. This is because, a normal route would traverse the Internet core (tier-1 + tier-2 ISPs) once while a consistent invalid route through 2 colluding adversaries traverses the Internet core twice (since the

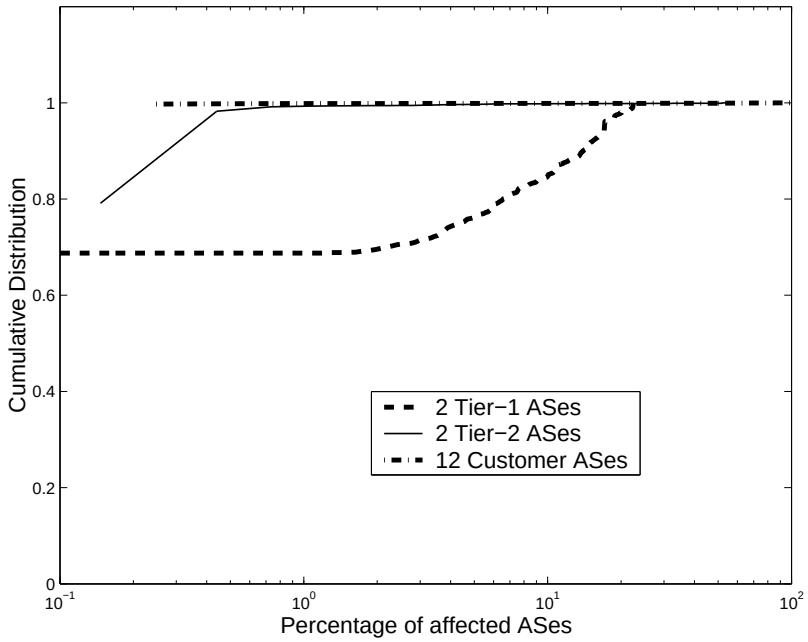


Figure 3.11: Effect of colluding adversaries with whisper + shortest path routing

adversary cannot remove any AS from the path). Hence, by choosing the shortest path we have a better chance of avoiding the invalid route. Figures 3.9, 3.10 and 3.11, illustrates this effect of colluding adversaries for 3 scenarios: (a) the current Internet with no protection; (b) whisper protocols with policy routing; (c) whisper protocols with shortest path routing. All these graphs show the cumulative distribution of the vulnerability metric (defined in Section 3.7.1) for a set of colluding malicious adversaries. We specifically consider three cases: (a) 2 colluding tier-1 ASs; (b) 2 colluding tier-2 ASs (c) 12 colluding customer ASs.

We make two observations. First, 12 randomly compromised customer routers can inflict the same magnitude of damage as that of two tier-1 nodes illustrating the effect of colluding adversaries in the current Internet. Typically, customer ASs are easier to compromise since many of them are unmanaged. Second, whisper protocols with shortest path routing drastically reduces the possibility of

colluding adversaries (in comparison to policy routing) propagating invalid routes without triggering alarms. In particular, even when 12 customer ASs are compromised, the effect on the Internet routing is negligible.

Whisper protocols with policy routing offers much lesser protection since BGP tends to choose routes based on the *local preference*. The typical policy convention based on stable routing and economic constraints is to prefer customer routes over peer and provider routes [54]. This preference rule increases the vulnerability of BGP to pick consistent invalid routes from customers over potentially shorter routes through peers /providers. In principle, this problem also exists in S-BGP. To strike a middle ground between the flexibility of policy routing and this vulnerability, we propose a simple modification to the policy engine: *Do not associate any local preference to customer routes that have an AS path length greater than* 2 (any route from a pair of colluding route should have a minimum path length of 3). We believe that this modification to BGP policies should have little impact on current operation since most customer routes today have a path length less than 3.

To summarize, whisper protocols in combination with the modified policies (emulating shortest path routing) can largely restrict the damage of colluding adversaries.

3.9 Discussion

We now discuss three specific issues not covered earlier.

Hijacking unallocated prefixes: With the deployment of Whisper, a malicious adversary can still claim ownership over unallocated address spaces without triggering alarms by propagating bogus announcements. One way of dealing with this problem is to request ICANN [3] to specifically

advertise unallocated address spaces with its own corresponding Whisper signatures whenever it notices an advertisement for an unallocated prefix. Additionally, to avoid a DoS attack on ICANN for such prefixes, routers should not maintain forwarding entries for these prefixes.

Route Aggregation: Whenever an AS aggregates several route advertisements into one, it is required to perform one of the following operations to maintain the consistency of the aggregated route: (a) Append the individual signatures corresponding to each advertisement so that an upstream AS can match at least one of the signatures with the whisper signatures for alternate routes to sub-prefixes. (b) If the AS owns the entire aggregated prefix (common form of aggregation in BGP), ignore the whisper signatures in the sub-prefixes and append its own whisper signature.

Other types of security attacks: Other than propagation of invalid routes, one can imagine other forms of routing attacks or misconfiguration errors which may result in routing loops, persistent route oscillations or convergence problems. Such problems are out of the scope of this work.

3.10 Summary

In this work, we consider the problem of reducing the vulnerability of BGP in the face of misconfigurations and malicious attacks. To address this problem we propose two techniques: Listen and Whisper. Used together these techniques can detect and contain invalid routes propagated by isolated adversaries, and a large number of problems due to misconfigurations. To demonstrate the utility of Listen and Whisper, we use a combination of real world deployment and empirical analysis. In particular, we show that Listen can detect unreachable prefixes with a low probability of false negatives, and that Whisper can limit the percentage of nodes affected by a randomly placed

isolated adversary to less than 1%. Finally, we show that both Listen and Whisper are easy to implement and deploy. Listen is incrementally deployable and does not require any changes to BGP, while Whisper can be integrated with BGP without changing the packet format.

Chapter 4

Reliable Communication in Unknown Networks

" Any fool can tell the truth, but it requires a man of some sense to know how to lie well. The best liar is he who makes the smallest amount of lying go the longest way."

– Samuel Butler, English writer

In this chapter, we will describe our theoretical result on the solvability of the reliable communication problem in unknown networks. We show that one can achieve reliable communication in an unknown network in the presence of k colluding adversaries if and only if the network has a minimum vertex connectivity of $2k + 1$. For the case of k independent adversaries that do not use any out-of-band mechanism to exchange information both prior and during protocol execution, we show that a vertex connectivity of $k + 2$ is necessary and sufficient. To put our result in context, we begin by describing the key results established in prior work in the context of known networks and

follow it up with a theoretical exposition of our result for unknown networks.

This chapter is organized as follows. In Section 4.1, we begin by summarizing the key prior results on reliable communication. In Section 4.2, we summarize our theoretical results on reliable communication. In Section 4.3, we describe path-vector signatures, a fundamental building block for achieving reliable communication. This path-vector signature is a more detailed version of the RSA-based Strong Split Whispers concept described in Chapter 3. In Section 4.4, we describe the reliable broadcast algorithm which enables every node to reliably broadcast its public-key to other nodes in the network provided the connectivity constraint is met. In this section, we also provide the proof of the reliable communication result for both the colluding adversaries case and the independent adversaries case. In Section 4.5, we analyze the complexity of the reliable broadcast algorithm. Finally, in Section 4.6, we describe the implications of this result and the brief summary of the chapter.

4.1 Prior results on reliable communication

The problem of reliable communication between nodes in the presence of byzantine adversaries [22, 23] was first considered in the context of the classic Byzantine General's problem [98, 76]. Consider a network $G = (V, E)$ where the edges in E represent reliable channels between nodes in V. By reliable channels, we mean channels over which message transmissions cannot be dropped, tampered, or forged. In the simplest case, when G is a clique, reliable communication between every pair of nodes can be trivially achieved. For a general graph G, Dolev [45] and Dolev *et al.* [46] proved that if there are k faulty nodes, then every pair of nodes can reliably communicate if

and only if the underlying graph is $2k + 1$ vertex connected. Biemel and Frankin [22] showed that the connectivity constraint can be relaxed if some pairs of nodes share authentication keys.

A simpler version of the reliable communication problem is the reliable broadcast problem where each node intends to reliably communicate the same message to every other node in the network. Reliable broadcast and reliable communication are dual problems of each other. Once reliable broadcast is achieved, one can perform pair-wise reliable communication by exchanging public keys through reliable broadcast.. The relationship between the reliable broadcast (RB) problem and the Byzantine agreement (BA) problem is summarized by the following observation:

Observation: Given n nodes of which k are adversarial, then two results hold: (a) BA $\implies$ RB; (b) If RB is achievable, then one can achieve BA if $n \geq 3k + 1$.

The first result implicitly follows from the fact that $\neg RB \implies \neg BA$. If two good nodes cannot reliably transmit messages between themselves, then they cannot achieve BA. The second result indirectly follows from previous works by Lamport *et al.* [76, 98] and Dolev [45].

4.2 Summary of our results

Given that the reliable broadcast and reliable communication problems are duals of each other, we focus on the reliable broadcast problem for the rest of this chapter. The primary result we prove in this chapter is:

Theorem 1. *Given a bound k on the number of adversaries, there exists a distributed algorithm Γ that achieves reliable broadcast in an unknown fixed-identity network $U(n, G, N)$ if and only if G is $2k + 1$ vertex connected.*

This result extends the prior result of Dolev [45] for unknown fixed-identity networks. Dolev proved that a minimum $(2k + 1)$ vertex connectivity is essential for achieving reliable broadcast even if the entire graph, G, is known to all the nodes. Our result shows that one can achieve reliable broadcast even in the case where G is unknown to the nodes provided the graph satisfies the $(2k+1)$ connectivity requirement. The time-complexity of the algorithm is dependent on the values of k, N and is discussed in detail in Section 4.5.

The fixed-identity assumption is critical towards addressing this problem. If this assumption is not met and an adversary uses different identities to different neighbors, then we can show prove the following result:

Lemma 1. *For any given integer $m > 0$, there exists an m-vertex connected network G on n nodes where each node is initially aware of the identities of only its neighbors, such that, a single adversary using multiple identities is sufficient to disrupt reliable broadcast in G.*

4.3 Path vector signatures

In this section, we describe the concept of *path vector signatures*, one of the basic building blocks we use to solve the problem of reliable broadcast. The signature mechanism described in this section is a detailed version of the RSA-based Strong Split Whisper signature mechanism presented in Chapter 3 (Section 3.4).

A path-vector signature is a signature associated with a message that traverses a particular path within the network. These signatures enable a good node to *differentiate* between genuine messages generated by good nodes from spurious ones generated by adversaries. An adversary (or a set of

adversaries) that intends to disrupt a good node v from reliably communicating a message $m(v)$, will attempt to propagate spurious messages $m'(v)$ claiming to be from v. To defend against such adversaries, we associate with each message a specific path-vector signature that is cryptographically computed and updated by every node along the path through which the message is propagated, so that no adversary can tamper with the message. Hence, an adversary intending to propagate a spurious message claiming to be from a good node v is forced to generate a *different signature* in comparison to the same message being generated by the source.

More formally, a *path-vector message* (m, s, p) consists of three parameters: a message m, the identity of the source s, and a path p containing the identities of the nodes the message traverses including the source s. A *path-vector signature*, $sgn(m, s, p)$, is a signature corresponding to a path-vector message (m, s, p) which is initiated by the source s and incrementally updated by every node along the path p. It is important to note that if a node u propagates a path-vector message (m, s, p) to v, then v's identity is already appended to the path p by u signifying that u has propagated the message to v. Hence, a node v that receives a message should have its identity as the last node in the path and cannot remove its identity (in case, v is an adversary). The path-vector signature, $sgn(m, s, p)$, should satisfy three properties:

1. **Verify:** Given (m, s, p) and $sgn(m, s, p)$, any node should be able to verify that the message traversed the nodes in path p provided the message m was initiated at s.
2. **Append an identity:** Let a node with identity x receives a message (m, s, p) along with $sgn(m, s, p)$. If x intends to forward the message to a neighbor with identity y, x should be able to compute the valid signature $sgn(m, s, p')$, for the message (m, s, p'), where p' is the path $(p, \{y\})$.

3. **Inability to modify:** Given a path-vector message, (m, s, p), an adversary *should not* be able to produce a valid signature for any message (m', s, p') where $m' \neq m$ or p' is not a path of the form (p, p_f) where p_f is any other path of identities. In other words, the adversary can append identities to the path but not remove identities.

We now discuss a simple path signature construction that satisfies these requirements. This construction relies on an underlying conventional public-key signature scheme G, where $G(m, P)$ refers to the message m signed using the public key P. There are several known schemes that can be used for this purpose, one example being the El Gamal signature scheme [53]. Consider a node v_1 sending a message m to node v_n over the path $(v_1, \ldots v_n)$. Let each node v_i generate a public key $g(v_i)$. The prescriptions of our protocol are as follows:

1. *Initialization:* Every node v_i generates its public key $g(v_i)$, and for $(i > 1)$, communicates it to its neighbor v_{i-1}.
2. *Message Initiation:* The source v_1 sends the message $m_1 = [\,(m, s, p_1), sgn_1\,]$ to its neighbor v_2 where $s = (v_1, g(v_1))$, $p_1 = [\,(v_1, g(v_1)), (v_2, g(v_2))\,]$, and $sgn_1 = \{G((m, s, p_1), g(v_1))\}$.
3. *Incremental update:* Node v_i ($i > 1$) receives message $m_{i-1} = [\,(m, s, p_{i-1}), sgn_{i-1}\,]$ from its predecessor v_{i-1}. It then sends message $m_i = [\,(m, s, p_i), sgn_i\,]$ to its successor v_{i+1} where $p_i = [\,p_{i-1}, (v_{i+1}, g(v_{i+1}))\,]$ and $sgn_i = \{sgn_{i-1}, G((m, s, p_i), g(v_i))\}$.

Note that each node v_i includes the identity of its successor v_{i+1} in its message m_i. This identity also includes the public key announced by v_{i+1}. Thus, in essence, the message received by node v_i consists of the original message m, the identity of originator s along with its claimed public key $g(v_i)$, and a path signature where the identity and public key of each hop is certified by its predecessor. Any node v_i on the path that receives a message (m, s, p) along with $sgn(m, s, p)$

can verify the correctness of each individual signature according to the signature scheme G. This construction satisfies the following lemma:

Lemma 2. *Any fake path vector message $M = (m, s, p)$ generated by an adversarial node with a genuine signature $sgn(M)$ can only be one of two categories:*

1. *M was generated by a single adversary v which generated fake public keys $g'(u)$ for all identities u that precede v in p.*
2. *Two colluding adversaries v, w that occur in p can insert a spurious path fragment comprising arbitrary identities (including identities of good nodes) between v and w in the path. For each such good node x whose identity was added by v and w, the adversarial nodes need to generate a fake public key $g'(x)$.*

It is essential for every node along the path to sign every message for this lemma to hold and distinguish any fake message. If not, an adversary can insert arbitrary path-fragments with identities and this can perturb the graph-computation process described in Section 4.4. This motivates the concept of a **keyed-identity** of a node denoted as $(x, g(x))$, where x is the identity and $g(x)$, the claimed public key of x in a message. Every path-vector message contains a string of keyed-identities. Any message with the keyed-identity $(x, g(x))$ is distinctly different from a message containing the keyed identity $(x, g'(x))$, of which, certainly one of the messages is bogus (since good nodes do not claim conflicting public keys). However, given only these two messages, a receiving node cannot immediately determine as to whether $g(x)$ or $g'(x)$ is the genuine public-key of x. We address this problem of determining the genuine keyed-identity in the next section.

4.4 Reliable Broadcast Algorithm

Based on the concept of path-vector signatures, we describe our reliable broadcast algorithm in this section. The algorithm uses two main ideas:

Keyed-identity graph computation: Every good node x uses the information from path-vector messages to continuously compute a keyed-identity graph G_x, where the nodes in the graph are of the form $(v, g(v))$, comprised of the actual identity v and the claimed public key $g(v)$. To prevent unnecessary path exploration, a path-vector message that contains no additional information(no new edge or vertex) is *not propagated further*.

Determining genuine identities: If the underlying graph G is $2k + 1$ vertex connected with k adversaries, then, between every pair of good nodes, there exists at least $k + 1$ vertex disjoint paths that traverse only good nodes. Hence, in the keyed-identity graph, G_x, if the number of *identity-disjoint* paths to a keyed-identity $(v, g(v))$ is at least $(k + 1)$, then x can conclude $g(v)$ to be the genuine public-key corresponding to identity v. Any bogus keyed identity $(v, g'(v))$ generated by adversaries can at most traverse k vertex disjoint paths since the identity of at least one adversary should be present in each path. If a message is not signed by every node along the path, an adversary can generate spurious edges and disrupt the computation of disjoint paths. The fact that adversaries can at most prove k disjoint paths to a fake node is critical for the solvability of this problem.

4.4.1 Asynchronous Broadcast Algorithm

Based on these two ideas, we describe an asynchronous algorithm to achieve reliable broadcast. Given a path-vector message (m, s, p) and its signature, we define the *keyed identity path* $P_I(m, s, p)$

associated with (m, s, p) to consist of vertices $(v_i, g(v_i))$, where v_i is the identity of a node in p and $g(v_i)$ is the public key of v_i in the signature. We use G_x to denote the keyed-identity graph computed by a node x with a set of neighbors $N(x)$. Every good node x performs the following set of operations.

BROADCAST(Node x, Neighbors $N(x)$)

1. *Asynchronous node wakeup:* A node can either begin broadcast by itself or begin transmissions upon receipt of the first message from a neighbor.
2. *Initiation:* G_x consists of one vertex $(x, g(x))$.
3. For every $u \in N(x)$, x transmits $(m(x), x, [x, u])$ to u along with its signature.
4. *Propagation:* For every path-vector message (m, s, p) with signature S that x receives from $u \in N(x)$, x performs:
   (a) *Immediate-neighbor key check:* Check if public-key of u in S matches the same public-key used in previous messages. If not, reject (m, s, p). If $v \in N(x) - \{u\}$ appears in p, then the public-key of v should also match the one directly advertised by v.
   (b) Verify S.
   (c) *Learn one vertex at a time:* Accept the message only if $P_I(m, s, p)$ contains at most one new keyed identity (at the end of the path) not present in G_x. If so, update G_x with $P_I(m, s, p)$.
   (d) *Message suppression:* If $P_I(m, s, p)$ adds no new vertices or edges to G_x, ignore the message.
   (e) To every $u \in N(x)$, x transmits (m, s, p') where $p' = p \cup \{u\}$ after updating the signature.

5. *Flow computation:* If the number of identity-disjoint paths to $(v, g(v))$ in G_x is at least $k+1$, then x deems v to be a genuine identity and $g(v)$ to be its public key. By identity disjoint paths, we mean that no two paths should contain two different vertices $(v, g(v))$ and $(v, g'(v))$ which share the same identity v.

The immediate-neighbor key check is necessary to ensure that if an adversary $v \in N(x)$, then v uses only a single keyed-identity $(v, g(v))$ in all its messages propagated to x. Any other message that x receives (from other neighbors) which contains the identity v is accepted only if it contains the same public key $g(v)$.

4.4.2 Proof of Theorem 1: asynchronous version

In this section, we will prove Theorem 1 and show that the asynchronous BROADCAST algorithm will eventually achieve reliable broadcast in an unknown network $U(n, G, N)$ if and only if G is $2k+1$ vertex connected. This proof consists of two parts. First, we establish the requirement that a minimum $2k+1$ vertex connectivity is necessary to achieve reliable broadcast. Next, we show how the BROADCAST algorithm achieves reliable broadcast.

Minimum (2k+1) connectivity requirement: Consider a graph H that is $2k$ vertex connected with k adversaries. Consider any vertex cut C of size $2k$ containing the k adversaries that separates H into two components A and B. The k adversaries can prevent nodes in A from reliably broadcasting to nodes in B by modifying every message $m(u)$ from $u \in A$ to $m'(u)$. Nodes in B cannot determine whether $m(u)$ or $m'(u)$ is genuine. Therefore, $2k$ vertex connectivity is insufficient to achieve reliable broadcast in the presence of k adversaries.

BROADCAST analysis: Let every good node execute the BROADCAST algorithm to broadcast $m(x)$. Consider a particular good node x. For every other good node v with public-key $g(v)$, if $(v, g(v))$ is a vertex in G_x, then x would have learnt the message $m(v)$ in the first path-vector message where $(v, g(v))$ is added to G_x since $(v, g(v))$ was the last node in that message.

Let $G_g = (V_g, E_g)$ represent the sub-graph comprising of all the edges between the set of good nodes. Given that G is $2k + 1$ vertex connected with at most k adversaries, G_g is at least $k + 1$ vertex connected. If every good node x can learn all the edges in E_g, then it can definitely compute $(k + 1)$ identity-disjoint paths to every other good node and hence, can successfully determine every other good node v, its public key $g(v)$ and message $m(v)$. To show that the BROADCAST algorithm achieves reliable broadcast, we need to show that every good node will learn all edges in E_g.

To prove that each good node will eventually learn E_g, let us separate the message exchange process between neighboring nodes into *rounds*. In round i, let each node exchange the new path-vector messages it learnt in round $i - 1$ with its neighbors. Within each round i, every node will learn all nodes within a distance i from the node. Hence, within $O(n)$ rounds every good node will learn all edges in E_g and hence discover all the genuine nodes in G. Finally, the following simple result on message complexity holds:

Lemma 3. *In an unknown network $U(n, G, N)$, let G comprise of e edges. When $k = 0$, each node transmits at most e path-vector messages to each neighbor using the BROADCAST algorithm.*

This result trivially follows from the message suppression step in the BROADCAST algorithm. This step ensures that for each edge in G, a good node propagates only one path-vector message. In the

absence of any adversary ($k = 0$), the number of path-vector messages along each link is bounded by e. In the presence of adversaries ($k > 0$) generating fake messages, the number of path-vector messages along each link depends on k. We discuss this complexity in Section 4.5.

4.4.3 Multiple Identities: Proof of Lemma 1

Consider the case where an adversarial node uses different identities to different neighbors. A single adversary using multiple identities is sufficient to disrupt reliable broadcast in certain types of networks however large the connectivity may be. For any positive integer value $m > 0$, one can construct a graph H consisting of $n(> m)$ vertices which satisfies the following two constraints:

1. H is $2m - 2$ vertex-connected but not $2m - 1$ vertex-connected.
2. H contains a vertex-cut C such that $m - 1$ vertices $A_1, \ldots A_{m-1} \in C$ have non-overlapping neighbor-sets $N(A_1), \ldots N(A_{m-1})$.

In essence if $A_1, \ldots A_{m-1}$ act as adversaries, then they can disrupt reliable broadcast in H. Given one such graph H, construct a new graph H' where the vertices $A_1, \ldots A_{m-1}$ are collapsed into a single vertex A whose set of neighbors represent the union of the set of neighbors of $A_1, \ldots A_{m-1}$. Clearly H' is m-connected. One can show that a single adversarial node A using different identities is sufficient to disrupt reliable broadcast in H'. To do so, the adversary A uses different identities to its neighbors in $N(A_i)$ for each $0 < i < m$, thereby creating an underlying graph that resembles H.

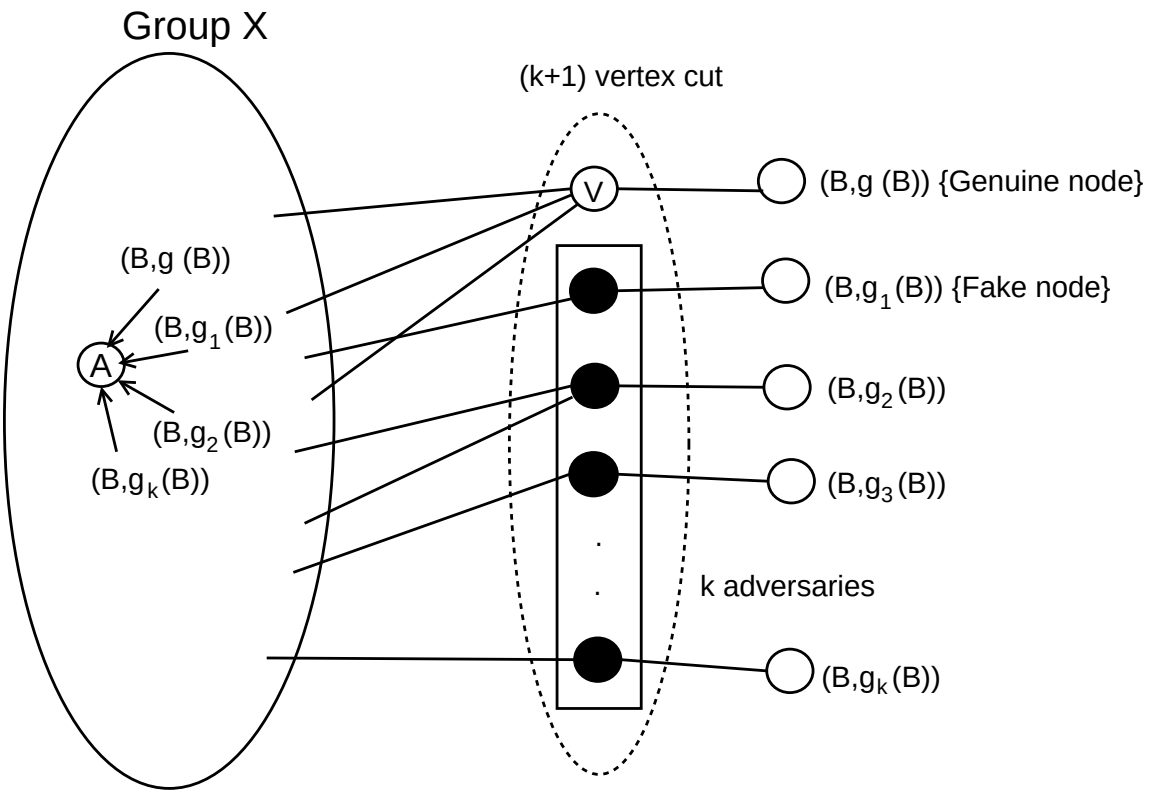


Figure 4.1: Independent adversaries case: In this example, a $(k+1)$ vertex cut containing k adversarial nodes separates node B from the rest of the nodes in the network. Here, node A receives conflicting claims for the keyed-identity for B and cannot determine as to which keyed-identity is the genuine one.

4.4.4 The Case of Independent Adversaries

Independent adversaries, by definition, do not communicate with each other before our during protocol execution. On the other hand, independent adversaries can potentially (in the worst case) target the same set of good nodes to affect. An additional assumption we explicitly make regarding independent adversaries is that, two independent adversaries are not direct neighbors in the underlying network enabling them not to directly exchange messages between them.

To handle the case of independent adversaries, we consider a simple modified version of the reliable broadcast algorithm presented earlier:

INDEPEDENTBROADCAST(Node x, Neighbors $N(x)$)

1. Perform all the steps of the BROADCAST(x, $N(x)$) algorithm except the final flow computation step.
2. *Modified Flow computation:* If the number of identity-disjoint paths to $(v, g(v))$ in G_x is at least 2, then x deems v to be a genuine identity and $g(v)$ to be its public key. By identity disjoint paths, we mean that no two paths should contain two different vertices $(v, g(v))$ and $(v, g'(v))$ which share the same identity v.

For the case of independent adversaries, we prove the following theorem:

Theorem 2. *Given a bound k on the number of independent adversaries, the INDEPENDENTBROADCAST algorithm achieves reliable broadcast in an unknown fixed-identity network $U(n, G, N)$ if and only if G is $k + 2$ vertex connected.*

Proof: The proof of this theorem follows directly from two observations. First, based on the proof

of Theorem 1, the BROADCAST algorithm guarantees that every node in the graph with discover all the edges between good nodes in the network. Given that the graph is $k+2$ connected with at most k adversaries, the subgraph of the good nodes is guaranteed to be 2-vertex connected. Hence, every good node will learn this entire graph and can determine 2-identity disjoint paths to every node in the graph. Therefore every node can determine the public key of every other good node in the network.

Second, to show that a good node does not learn any fake node as a genuine node follows from the definition of independent adversaries. Two independent adversaries that propagate an incorrect announcement about the same identity X will generate two different keyed-identities for X. By definition, independent adversaries do not communicate either during or prior to protocol execution. Hence, the keyed identities generated for the same identity are different. Therefore, if $(X, g'(X))$ is a fake keyed-identity generated by an adversary M, the number of identity disjoint paths to $(X, g'(X))$ is at most 1 since all such paths have to traverse Y.

To complete the proof, we need to show that a minimum connectivity of $k+2$ is essential. We show this by construction. Consider any graph G which is $k+1$ vertex connected where the set of k adversaries along with an additional node V form a $k+1$ node vertex cut (as illustrated in Figure 4.1). In such a graph, there exist two nodes A and B which have only one vertex disjoint path between them through the good node V. When each of the k independent adversaries generates a spurious announcement with a modified keyed identity for B, A cannot determine the true identity of B since it has at most 1 identity disjoint path to each keyed-identity of B. Hence, the different keyed-identities of B appear indistinguishable. In this graph, one cannot achieve reliable communication in the presence of k independent adversaries. $\square$.

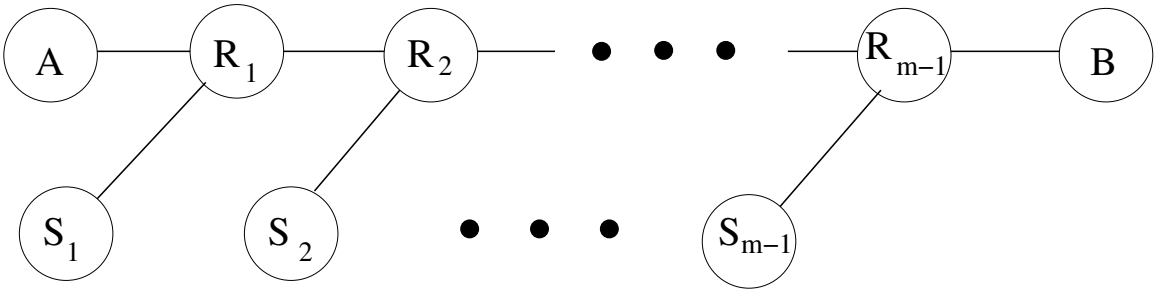


Figure 4.2: Packet delay along a linear path

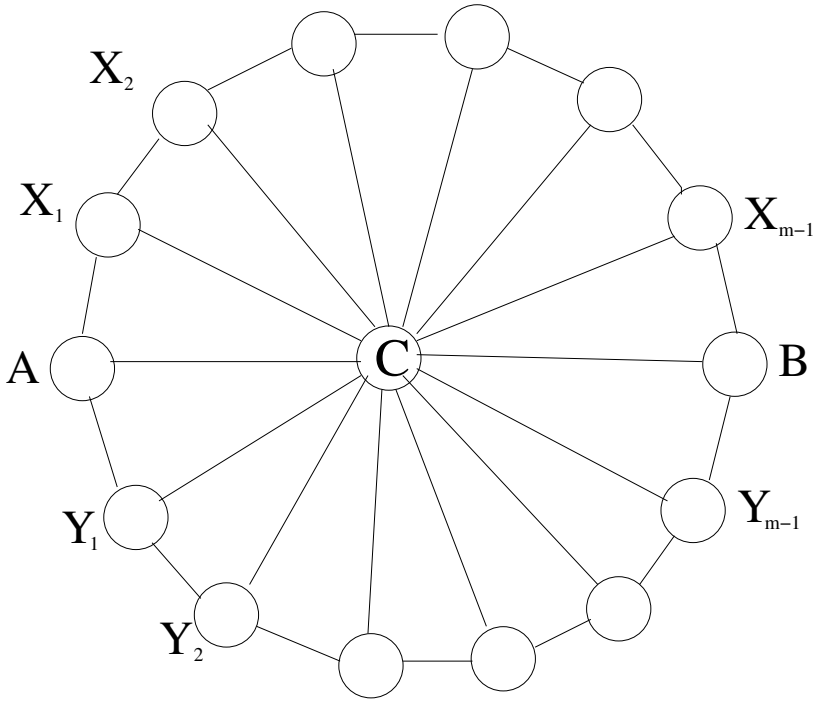


Figure 4.3: Example topology for G with $2m + 1$ nodes with C being an adversarial node.

4.5 Complexity analysis

In this section, we consider a synchronous version of the BROADCAST algorithm and analyze its time complexity. One needs to be careful when analyzing the time-complexity of an algorithm in an unknown fixed identity network. The traditional Byzantine agreement literature uses the concept of rounds [45, 76, 52, 78] to analyze time-complexity. However, given that the entire network is unknown, enforcing the global concept of a round is not feasible. On the other hand, a completely asynchronous mode of communication [24, 78, 30] is also not suitable for our analysis since it is not possible to provide time guarantees for message deliveries. Hence, we revert to the traditional synchronous model and *locally* enforce the concept of time by imposing capacity constraints on links based on the following definition:

Definition A network G is said to be *capacity-constrained* if every node can only transmit $O(1)$ bits of information to each of its neighbors in a single unit of time.

Capacity constraints enable us to loosely enforce the concept of a round globally while every node operates locally at its own link-capacity rates. By enforcing an $O(1)$ capacity constraint on each link, we ensure that the ratio of the time to deliver a message along a link is $O(1)$. While we analyze the time complexity of our algorithms based on the capacity-constrained assumption, it is conceivable that alternative measures of time may apply.

4.5.1 Message scheduling algorithm

To produce a synchronous version of the BROADCAST algorithm for a capacity-constrained network, it is essential to determine the mechanism used to schedule messages at every node. In a

capacity constrained network, each node receives multiple messages from each neighbor but can propagate only one message on each outgoing link. Hence, each node needs to buffer messages and use a scheduling algorithm to prioritize the messages to be transmitted. The lemma described below shows that using a simple FIFO scheduling algorithm does not suffice:

Lemma 4. *If every node uses a simple First-in-First-out (FIFO) queue with an infinite buffer to schedule messages on each link, there exists an unknown network, $U(n, G, N)$, where G is $3-$connected such that the minimum time complexity of any algorithm to achieve reliable broadcast in the presence of a single adversary is $2^{(n-3)/2}$.*

Proof: We first show that in the particular topology illustrated in Figure 4.2, the delay incurred in transmitting a single message from A to B separated by m capacity-constrained hops can be as high as 2^{m-1} if intermediary nodes use FIFO queues. Let this topology be capacity constrained in that every node can transmit only one message along every link in unit time. Let all nodes begin transmission at time $t = 0$ with all queues initially being empty. Now, assume that the nodes S_i connected to R_i continuously transmit one message every unit time. In this case, if all R_i's use FIFO queuing, a single message from A to B will incur a worst-case delay of 2^{m-1}. Since each S_i transmits one packet per unit time to R_i, in the worst-case, A's packet reaches R_2 from R_1 at time $t = 2$, reaches R_3 at $t = 4$ and reaches R_{i+1} at $t = 2^i$. Hence, the bound 2^{m-1}.

Next, using the previous result we construct a topology where reliable broadcast with FIFO queues has a minimum time complexity of $2^{(n-3)/2}$. Consider a modified topology (as shown in Figure 4.3) with $2m + 1$ nodes comprising of a loop of $2m$ nodes and a central node C connected to all the nodes in the loop. Now let node C be the only adversarial node that continuously injects fake path-vector messages on each of its links. Each such fake message is generated from a non-existent

vertex and also contains an arbitrary non-existent path. Now, two good nodes A and B can only communicate through the two paths in the loop of length m. By the previous argument, if all nodes use a FIFO queue with an infinite buffer, any message from A to B will incur a minimum delay of 2^{m-1}. However, any algorithm that achieves reliable broadcast should enable at least one message to be communicated from A to B. Hence, the minimum time complexity of any such algorithm using FIFO queues is lower bounded by $2^{m-1} = 2^{(n-3)/2}$. $\square$

The above argument can also be directly extended to the Fair Queuing discipline where a node divides a link's capacity equally amongst all its other neighbors. In the above example, FIFO scheduling and Fair queuing does not work primarily because an adversarial node can flood the network with spurious messages and delay the delivery of packets.

Identity-based rate limiting

The goals of identity-based rate-limiting are two-fold: (a) hold nodes accountable for every message they transmit and limit the capability of adversaries to flood messages; (b) associate a higher priority to the messages from lesser-known nodes which have not received enough opportunities to transmit messages. To achieve these, the key idea is to *rate-limit messages* across identities and keyed-identities. To do so, the algorithm computes two simple metrics:

1. *Identity priority:* Let $I_1(u, t)$ represent the number of path-vector messages transmitted prior to time t with identity u in the path. A message (m, s, p) has an identity priority $p_1(m, s, p)$ which is the maximum value of $I_1(u, t)$ for all identities $u \in p$.
2. *Keyed-identity priority:* Let $I_2((u, g(u)), t)$ represent the number of messages transmitted prior to time t with keyed identity $(u, g(u))$ in the keyed-identity path. The keyed-identity

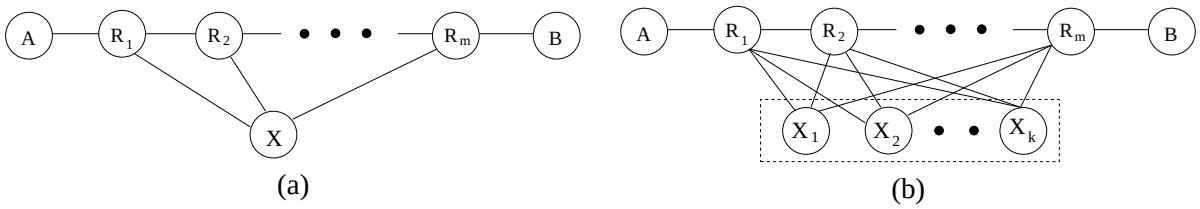


Figure 4.4: (a) Single adversary case. (b) Example topology for the case when $k > 1$.

priority $p_2(m, s, p)$ of a message is the maximum value of $I_2((u, g(u)), t)$ over all keyed identities $(u, g(u))$ in the path.

The scheduling strategy is as follows:

1. Schedule message (m, s, p) with the lowest identity priority, $p_1(m, s, p)$.
2. If multiple messages have the same minimum identity priority, schedule the message among them with the lowest keyed-identity priority, $p_2(m, s, p)$.
3. Use FIFO as a final tie-breaking rule.

The rationale behind maintaining two separate priorities is two-fold. First, the *identity-based priority* is essential to prevent an adversary to use multiple keyed identities and propagate spurious messages. For example if we use only keyed-identity priorities in the topology illustrated in Figure 4.2(b), the lower bound argument used in the case of FIFO holds in this case too. Second, the keyed-identity based count is essential because an adversary can artificially increase the identity-based count of a good node by inserting the good node in a path. However, an adversary cannot artificially increment the keyed-identity based count of a node.

4.5.2 BROADCAST algorithm complexity

We use $G_g = (V_g, E_g)$ to represent the sub-graph of G comprising of only the good nodes V_g and E_g, the set of edges between them. Assuming G_g is connected, let $diam(G_g)$ denote its diameter. We prove the following theorem on the BROADCAST algorithm.

Theorem 3. *Given a bound k on the number of adversaries and a bound Δ on $diam(G_g)$ in an unknown network $U(n, G, N)$, the BROADCAST algorithm with identity-rate limiting (IRL) achieves reliable broadcast in:*

1. $O(N^2 n \log n)$ *time for* $k = 0$.
2. $O(N^3 n \log n)$ *time for* $k = 1$.
3. $O((k+1)^{\Delta} N^2 n \log n)$ *time for* $k > 1$.

Proof. In a capacity constrained network, a node can transmit only $O(1)$ bits per unit time. For simplicity, we bound the total number of bits consumed by every path-vector message by $O(n \log n)$ ($\log n$ bits for each identity and n for the maximum number of signatures in a message). We analyze the complexity separately for three cases:

Lemma 5. *When $k = 0$, BROADCAST + IRL achieves reliable broadcast in $O(ne \log n)$ time where e is the number of edges in G.*

Proof: If the first node initiates transmission at time $t = 0$, every node will receive a wake up signal after n path-vector message transmissions given that the diameter of G is bounded by n. After wakeup, the message suppression step in the BROADCAST algorithm ensures that every node learns G after e message transmissions along each edge. Since each node will always have

a new path-vector message to transmit and that each message has $O(n \log n)$ time complexity for transmission on a link, BROADCAST requires $O(en \log n)$ time to converge. $\square$

Lemma 6. *When $k = 1$, BROADCAST+IRL achieves reliable broadcast in $O(n^2 e \log n)$ time.*

Proof: We first analyze a simple scenario as illustrated in Figure 4.4(a) where nodes A and B are $m + 1$ hops away and A propagates one message to B in the presence of an adversary X that continuously injects packets along each hop in the path (No other node propagates any message in this example). If all transmissions begin at $t = 0$ with the identity based counts set to zero, the IRL algorithm will ensure that for every message that A propagates, X can at most insert m messages, one along each of the links XR_i. Hence, A's message is delivered to B within $(m+1)n \log n$ time assuming a simple upper bound of $n \log n$ bits for each message.

In the general case, from the proof of Lemma 5, we know that every good node needs to receive e different path-vector messages to completely learn the good graph G_g. To analyze the worst-case time bound, we analyze each such message in isolation. Each message (m, s, p) that traverses a path p comprising of $|p|$ good nodes experiences a delay of $(|p| + 1)n \log n$ since an adversary X can inject spurious messages (either directly or indirectly) for each node along p. $|p|$ represents the length of path p and is clearly bounded by $n - 1$. Hence, each message in isolation if delivered one at a time is delivered to a good node in $O(n^2 \log n)$ time. The time to deliver e messages is bounded by $O(n^2 e \log n)$ time. $\square$

Lemma 7. *For a general value k, given a bound $\Delta \geq diam(G_g)$, using BROADCAST+IRL, every good node achieves reliable broadcast after $O((k+1)^{\Delta} en \log n)$ time.*

Proof: Much like Lemma 6, we consider a simple scenario of two good nodes A and B separated

by m hops (as shown in Figure 4.4(b)), except that the adversaries $X_1, \ldots, X_k$ continuously inject messages at every node R_i along the path. Unlike Lemma 6, we show that the delay incurred by every message from A in this case can be as high as $O((k+1)^m \times n \log n)$. The crux of the argument is that at each hop, the adversaries can delay the packet at most by a factor $(k+1)$. In this case, every adversary uses a different keyed identity with respect to each router along the path. Therefore, for every packet that R_{j-1} transmits to R_j, each of the adversaries can append one more packet to the queue at R_j creating a delay factor of $k+1$. Hence A's packet reaches B after $(k+1)^m$ packet transmissions. Each packet transmission takes $O(n \log n)$ time providing an overall complexity of $O((k+1)^m \times n \log n)$. In the general case, given a bound Δ on $diam(G_g)$, the information about every edge in G_g can be transmitted along a path of at most Δ hops. Hence, the maximum time required to discover e edges in G_g is bounded by $O((k+1)^{\Delta} en \log n)$. □

Stoppage constraint: The final element of the proof is the stoppage constraint. Since no node is aware of the values of n and e, each good node cannot determine when the full graph is learned. We use the bound N to represent a bound on n and N^2 as a bound on e, the number of edges. These bounds need to be applied only on the number of messages but not on the size of each message. Hence, we can still retain the $O(n \log n)$ bound on the maximum time complexity of a single message. Replacing these bounds in Lemmas 5, 6 and 7 completes the proof. □

4.5.3 Lower-bound time complexity

We show the following lower-bound on the time complexity of any reliable broadcast algorithm.

Lemma 8. *Given a bound k on the number of adversaries, there exists a capacity constrained network $U(n, G, N)$ where G is $2k+1$ vertex connected, such that any algorithm that achieves*

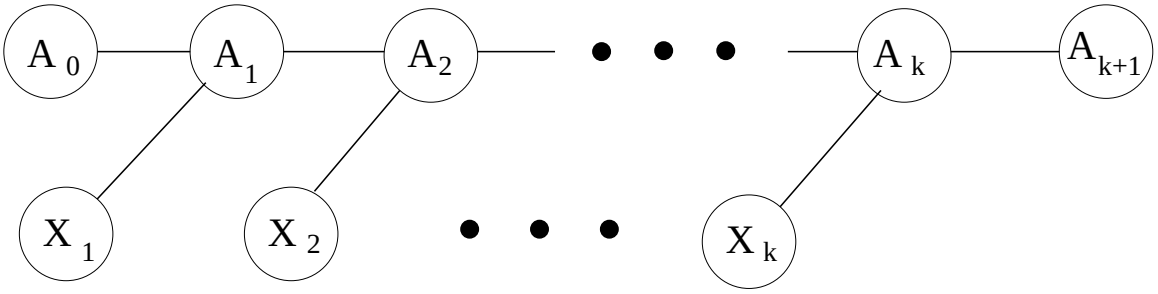


Figure 4.5: Single path with k adversaries.

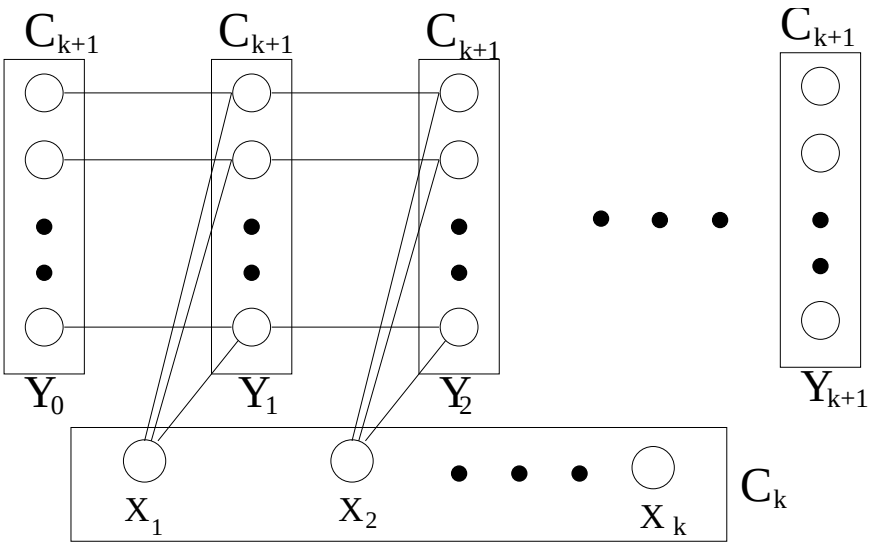


Figure 4.6: Example topology for Lemma 1. C_k and C_{k+1} denote cliques of size k and $k+1$.

reliable broadcast in $U(n, G, N)$ has a minimum time complexity of 2^k.

*Proof:*To illustrate the lower bound, consider the topology illustrated in Figure 4.5 where nodes A_0 and A_{k+1} are connected by a path of length $k+1$ and the k adversaries $X_1, \ldots X_k$ directly connect to the nodes along the path. In this example topology, the adversaries can delay the propagation of a single message from A_0 to A_{k+1} by a minimum time of 2^k.

For simplicity, we assume that the propagation of a single message takes unit time along a link. The basic idea of the argument is that for every message that A_{i-1} transmits to A_i, X_i can transmit a message to A_i. Since A_i cannot differentiate a message from A_{i-1} and X_i, it has to accord both messages equal priority. Hence, for a single message from A_0, X_1 can transmit one message thereby forcing A_1 to propagate 2 messages to A_2. Extending the argument, A_{i-1}, in the worst case, propagates 2^{i-1} messages to A_i to finally propagate a single message from A_0. During this period, X_i can propagate an equal number of message delaying A_0's message to A_{i+1} by time 2^{i+1}. Hence, in the worst case, A_0's message needs 2^k time to reach A_{k+1}.

Based on this example, consider the graph H illustrated in Figure 4.6, where every node A_i (in the previous figure) is replaced with a clique Y_i of size $k+1$ and each clique is connected to the next with a matching of size $k+1$. The adversarial nodes $X_1, \ldots X_k$ form a complete clique. H is $2k+1$ connected with k adversaries. As in the previous example, let X_i continuously inject bogus messages to all nodes in the clique Y_i. To achieve reliable broadcast in H, every node in Y_0 needs to propagate at least one message to nodes in Y_{k+1}. The previous argument can be extended to show that this message transmission from Y_0 to Y_k can be delayed by 2^k in the worst case. □

Notice the wide gap between the lower-bound result and the time complexity of our algorithm.

Addressing this complexity gap is an open research problem; our work primarily illustrates the existence of an algorithm to achieve reliable broadcast but does not target optimality.

4.5.4 Limitations of capacity-constraints

Analyzing the time complexity of a distributed algorithm with capacity constraints on link-topologies is a very restrictive model. Lemma 4 illustrates the limitation of FIFO in the face of a single adversary, where a single message transmission can be exponentially complex with capacity constraints. In other words, many simple distributed algorithms including the emulation of a single round in a Byzantine agreement algorithm has exponential complexity. Therefore, in this restrictive model, several simple distributed algorithms end up having exponential complexity in the face of adversarial nodes. Given this, the bound for the case $k > 1$ of the BROADCAST algorithm should be viewed as a worst-case bound (where an adversary generates infinite bogus message) which is not completely reflective of the real-life scenario. In a realistic setting, we would expect the asynchronous version of the BROADCAST algorithm to have much lower run-time complexity and anticipate an adversary to generate only a finite number of bogus announcements. With finite bogus announcements from adversaries, the complexity is polynomial in the number of nodes with capacity constraints. In realistic settings, we anticipate adversarial nodes to propagate only a finite number of bogus announcements.

4.6 Implications and summary

In this chapter, we proved three important results on the reliable communication problem in unknown networks:

1. To achieve reliable communication in the face of k colluding adversaries, we require a minimum vertex connectivity of $2k + 1$.

2. To achieve reliable communication in the face of k independent adversaries, a minimum vertex connectivity of $k + 2$ suffices.

3. The fixed identity criterion is critical to achieve reliable communication. If this is not met, a single adversary is sufficient to disrupt reliable communication.

Achieving reliable communication in unknown networks has two important practical implications. First and foremost, the reliable broadcast algorithm described in this chapter enables every good node to distribute its public key to every other good node in the network. In essence, this algorithm provides decentralized key distribution between good nodes which is what a PKI strives to provide. After this step is accomplished, one can translate PKI-based solutions into purely decentralized security solutions. Second, addressing the problem in the context of unknown networks is essential to apply these techniques for securing routing protocols. In Chapter 6, we describe the system implementation of the reliable communication toolkit which uses these basic algorithms and exports a generic set of security primitives that can be used by a variety of routing protocols.

This chapter focused on the constraints under which reliable communication is achievable. In the next chapter, we consider the case of *sparse networks* where the connectivity constraint is not met.

Chapter 5

Reliable Communication in Sparse Networks

“ Scott says: ‘Fool me once, shame on you. Fool me twice, shame on me!’. Chekov retorts: ’I know this saying. It was invented in Russia.’ ”

– In “Star Trek: Chekov’s Russian Pride”

In this chapter, we address an alternative aspect of the reliable communication for sparse networks which do not satisfy the $(2k+1)$ vertex connectivity requirement. In such networks, it is fundamentally impossible to achieve reliable broadcast. In such networks, we show that it is possible to *limit the damage* that an adversaries may cause in sparse networks. We specifically study the reliable broadcast in sparse networks for the specific case of a single adversary ($k = 1$) and show optimality results for this case. While the techniques we describe in this chapter are also applicable to the case for multiple adversaries, it is an open research problem to determine the optimal defense strategy

for such networks.

The rest of this chapter is organized as follows. In Section 5.1, we summarize the two key theoretical results presented in this chapter: (a) optimal defense mechanism for sparse networks; (b) the amount of damage that an adversary can cause in power-law random graphs (Internet-like graphs) is very small. In Section 5.2, we elaborate on the optimal defense strategy and in Section 5.3, we summarize the proof of the power-law random graph result. Finally, we summarize the key results presented in this chapter and their implications in Section 5.4.

5.1 Summary of our results

In this chapter, we only focus on the case of a single adversary *i.e.,* $k = 1$. Hence, we only deal with cases where the connectivity of the underlying network is $1-$connected and $2-$ connected where the network does not permit reliable broadcast in the presence of a single adversary *i.e.,* $k = 1$. In $3-$connected networks, one can achieve reliable communication in the presence of a single adversary. In the rest of the chapter, we use the term *sparse network* to refer to $1-$connected and $2-$connected networks.

5.1.1 Defining the minimum damage caused by an adversary

Given that reliable communication problem is not achievable in a sparse network, an adversary can create a certain minimum amount of damage. To motivate this, consider the example topology illustrated in Figure 5.1 where the adversary x and the node y form a 2-vertex cut, where they separate the rest of the nodes into two groups: group A and group B. In this case, x can prevent

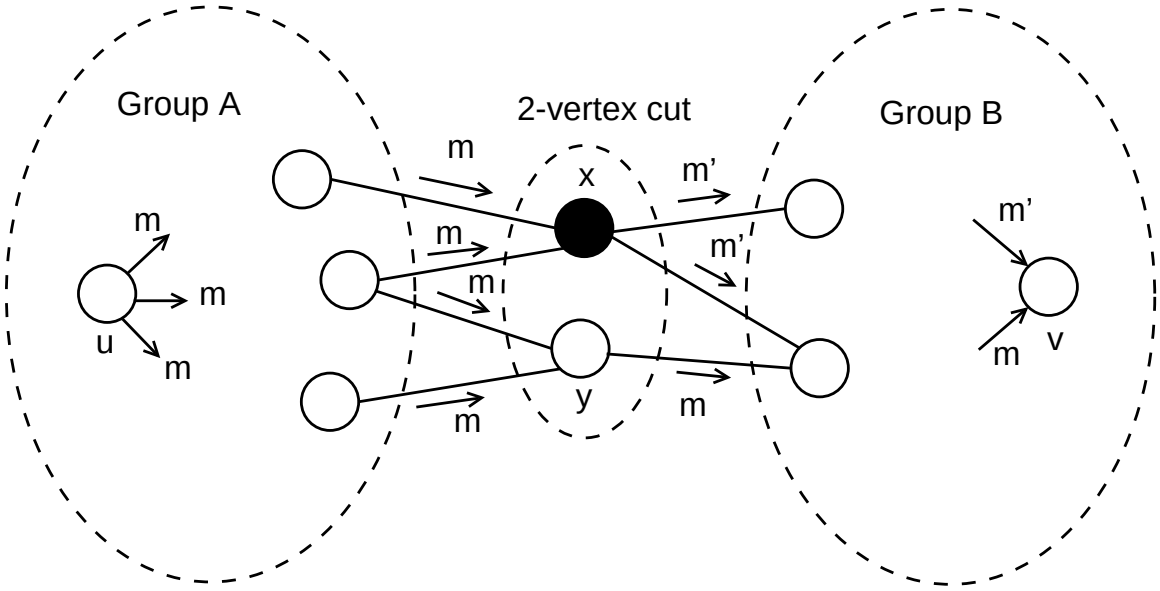


Figure 5.1: Example of a sparse network where a single adversary x can disrupt reliable communication between two groups of vertices A and B.

nodes in group A from reliably communicating with nodes in group B. Hence, the minimum damage that x can create in this network is to separate the set of good nodes into two reliable communicating groups $A \cup \{y\}$ and $B \cup \{y\}$ where two nodes in different groups cannot reliably communicate with each other (unless it represents the same node in two groups). In general, we can define a *broadcast partition* as follows:

Definition An adversary is said to create a *broadcast partition of size* m, if it can classify the set of good nodes, V_g, into m groups $(X_1, \ldots X_m)$, such that the following constraints are met:

1. Each X_i is non-empty and $V_g = X_1 \cup \ldots \cup X_m$.
2. Nodes within X_i can reliably broadcast to other nodes in X_i but not to nodes in X_j for $j \neq i$.

One dimension for quantifying the minimum damage that an adversary can cause is based on the *maximum size of the broadcast partition* the adversary induces. However, this may depend on the

location of the adversary in the topology and also the degree of the adversary in the underlying topology. An alternative dimension for quantifying the damage of an adversary is to measure the *cumulative damage* that an adversary may cause based on the following definition:

Definition Given a broadcast partition $B = (X_1, \ldots X_m)$ of V_g created by an adversary w, the *cumulative damage* caused by w is given by $min_i|V_g - X_i|$. If X_j represents the largest broadcast group in B, then the cumulative damage represents the number of nodes that cannot communicate with X_j.

The cumulative damage of an adversary measures the number of nodes that a single adversary can affect. If we consider Internet routing as an example, this metric represents the lower bound on the number of nodes that an adversary can affect by propagating incorrect messages.

5.1.2 Our theoretical results

We prove the following result for reliable broadcast on sparse networks:

Theorem 4. *Given that an unknown fixed-identity network $U(n, G, N)$ has exactly a single adversary A with degree $d(A)$, the following statements hold:*

1. *If G is $1-$connected, A can create a broadcast partition of size at most $2 \times d(A)$.*
2. *If G is $2-$connected, A can create a broadcast partition of size at most $d(A)$.*

These results are optimal in the sense that these represent not only the lower-bound on the amount of damage that an adversary can cause but also a tight upper-bound. We provide an algorithm that restricts the amount of damage an adversary can cause and achieves the lower bound. While Theo-

rem 4 provides only a bound on the size of a broadcast partition, it does not provide a bound on the cumulative damage caused by a single adversary. One can construct $1-$connected and $2-$connected graphs where the cumulative damage is $O(n)$. However, we show that the cumulative damage of an adversary in a power-law random graph [18, 39] is much smaller than n as summarized below:

Theorem 5. *Given an unknown fixed-identity network $U(n, G, N)$ where G is a power-law random graph on n vertices with parameter α satisfying $2 < \alpha < 3$, the cumulative damage caused by a single adversary is bounded by $O(n^{1/\alpha} \times (\log n)^{(5-\alpha)/(3-\alpha)})$ with high probability.*

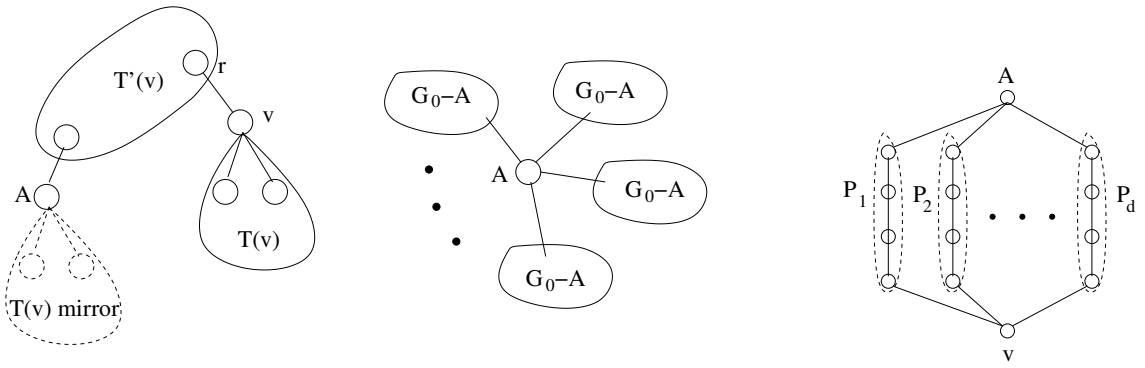
This result has important practical implications for the Internet and many social networks that exhibit structural similarities to power-law random graphs [18]. In such types of networks, the cumulative damage that an adversary may cause is very small compared to the size of the network.

5.2 Sparse networks: Proof of theorem 4

In this section, we describe our results for the problem of reliable broadcast in $1-$connected and $2-$connected graphs in the presence of a single adversary *i.e.,* $k = 1$.

5.2.1 Lower bound of Theorem 4

In this section, we show the existence of $1-$connected and $2-$connected graphs such that a single adversary with a degree d can create broadcast partitions of sizes $2d$ and d respectively in these graphs. This establishes the lower bound on the size of a broadcast partition that a single adversary can create in 1-connected and $2-$connected graphs. In this analysis, we assume that the adversary is aware of G while good nodes are aware of only their neighbors. This assumption is valid since



(a) Graph G_0 (b) d replicas of $G_0 - A$ (c) 2−connected graph example

Figure 5.2: Lower bound analysis for sparse graphs: (a) An example tree G_0 with adversary A being a leaf node. (b) Example tree with adversary A having degree d. (c) Example 2-connected graph.

the adversary can first learn about G from good nodes and then propagate spurious messages. The following lemmas establish the lower-bound.

Lemma 9. *There exists a tree G such that an adversarial node A in G with degree d can create a broadcast partition of size $2d$ in $U(n, G, N)$.*

Proof: We first analyze the case when $d = 1$. Let G_0 be a tree rooted at a node r (refer Figure 5.2(a)). Let v be a child of r such that the sub-tree rooted at v is non-empty and does not contain the adversarial node A. Let $T(v)$ represent the set of nodes in the sub-tree rooted at v (excluding v) and let $T'(v) = V - T(v) - \{v, A\}$. Pick any tree G_0 such that $T(v)$ and $T'(v)$ are non-empty. For every node $u \in T(v)$ that attempts to reliably broadcast a message $m(u)$, let A propagate a spurious message $m'(u)$ as if u propagated the message (as illustrated in Figure 5.2(a)). Since no node in $T'(v)$ is directly adjacent to any node in $T(v)$, these nodes will receive two messages $m(u)$ and $m'(u)$ and cannot figure out which message is genuine. All genuine messages from $T(v)$ are routed through v and all spurious messages $m'(u)$ are routed through A and nodes in $T'(v)$ cannot determine which node to believe. Hence no node in $T'(v)$ can reliably communicate with any node

in $T(v)$. Thereby, A creates a broadcast partition $(T(v) \cup \{v\}, T'(v) \cup \{v\})$ of size 2.

For the general case, we can create a tree G with d replicas of G_0 with the vertex A merged across all these replicas (as illustrated in Figure 5.2(b)). In this case, by simply dropping all good messages traversing it, A can create d separate components which cannot reliably broadcast to each other. Within each such component, A acts as a leaf node and can create a broadcast partition of size 2. Hence, A can create a broadcast partition of size $2d$. □

Lemma 10. *There exists a 2-connected graph G such that an adversarial node A in G with degree d can create a broadcast partition of size d in $U(n, G, N)$.*

Proof: Consider the graph G illustrated in Figure 5.2(c), where $P_1, \ldots P_d$ are paths of good nodes such that the end-points of each path connect to two separate vertices A and v. In this case, A has degree d. For every node $u_i \in P_i$ that intends to reliably broadcast a message $m(u_i)$, A propagates a message $m'(u_i)$ to all its neighbors in P_j where $j \neq i$. Hence, every node $u_j \in P_j$ will receive two messages: $m(u_i)$ (through v) and $m'(u_i)$; hence for every $u_i \in P_i$ ($i \neq j$), u_j cannot ascertain which of the two messages is genuine. Therefore, no pair of nodes (u_i, u_j) , $u_i \in P_i$ and $u_j \in P_j, j \neq i$, can reliably communicate. A creates a broadcast partition ($P_1 \cup \{v\}, \ldots, P_d \cup \{v\}$) of size d. □

5.2.2 Upper bound analysis of Theorem 4

Now, we describe an optimal penalty-based defensive strategy that a good node uses to limit the size of a broadcast partition that an adversary can cause to the lower bound. This mechanism works only for $k = 1$. This defense strategy is a generalization of the penalty-based filtering strategy described

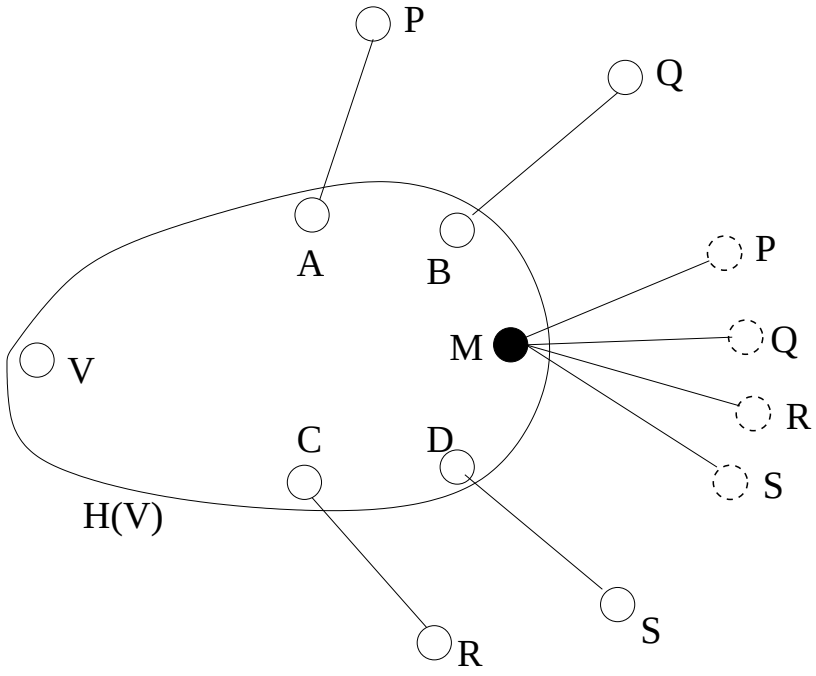


Figure 5.3: An example topology for the penalty based defense strategy.

in Chapter 3 (Section 3.4). The defensive strategy uses one key corollary that directly follows from Theorem 1 (described in Chapter 4):

Corollary 1: *Let H be a $2-$connected subgraph of G in an unknown network $U(n, G, N)$, comprising of only good nodes. In the presence of a single adversary, M, every node in H can reliably broadcast to all nodes in H.*

This corollary follows directly from the BROADCAST algorithm described in Chapter 4. Using this algorithm, every node in H can reliably discover two identity disjoint paths to every other node in H. Therefore, every node in H can reliably broadcast its public key to all other nodes in H.

We now explain the intuitive idea behind the penalty-based defense strategy using an example illustrated in Figure 5.3. Let $H(v)$ represent the set of all keyed-identities that have a 2 identity disjoint

paths to v. An adversary M cannot disrupt any of these nodes. In this example, however, M attempts to disrupt reliable communication to multiple nodes P, Q, R, S which connect (directly or indirectly) using good paths to different nodes A, B, C and D in $H(v)$. From the perspective of v, M and A disagree on P, M and B disagree on Q, M and C on R and finally M and D on S. Hence, if v associates a penalty with a node for every identity-disagreement, then M obtains the maximum penalty of 4 while the nodes A, B, C, D each obtain a penalty of 1. Therefore, v notes M to be a suspicious candidate and filters M's messages and chooses the genuine identities propagated by A, B, C, D.

Now, we present the penalty-based algorithm based on the idea explained above. Initially, every node x executes the BROADCAST+IRL algorithm for the case of $k = 1$ and computes the keyed-identity graph G_x. Next, they apply the following penalty algorithm on G_x.

PENALTY (Node x, Graph G_x)

1. We declare a keyed-identity $(y, g(y))$ to be *genuine* if x has two identity disjoint paths to $(y, g(y))$.
2. The *tail-end* of a keyed-identity $(v, g(v))$ is the path from the node $(y, g(y))$ to $(v, g(v))$ in G_x such that $(y, g(y))$ is the last genuine node in any path from x to $(v, g(v))$.
3. An identity u has a *conflict* if there are at least *two* keyed identities with the same identity in G_x.
4. **Penalty assignment:** The *penalty* of an identity u is the number of distinct conflicting identities y for which some keyed-identity $(u, g'(u))$ appears in the tail-end of a keyed-identity $(y, g(y))$.

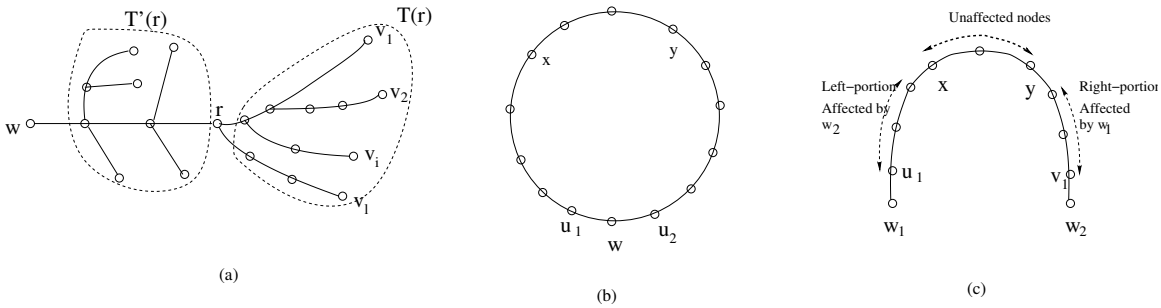


Figure 5.4: (a) Example topology for analysis for trees. (b) Example of a cycle with adversary A. (c) Affected and unaffected regions in a cycle due to adversary A.

Node-selection criteria: Based on the penalties assigned to identities, the criteria for selecting nodes is simple. For each keyed identity, $(v, g(v))$ that has a conflict, determine the *maximum penalty* of the identity u that appears in the tail-end of $(v, g(v))$. Choose the keyed-identity with the *minimum value of the maximum penalty*. If no unique minimum exists, then we declare the identity as non-identifiable.

Revisiting the example in Figure 5.3, v will notice P, Q, R and S to be conflicting identities. The genuine keyed-identity of P propagated by M will have a maximum penalty of 1 while the fake keyed-identity generated by the adversary M will have a penalty of 4. Hence, the penalty based algorithm chooses the genuine keyed-identities for P, Q, R, S.

The following lemma holds regarding the penalty-based algorithm:

Lemma 11. *In an unknown network $U(n, G, N)$, if all good nodes use the BROADCAST+IRL+PENALTY algorithm for determining genuine nodes, a single adversary with degree $d(A)$ can create a broadcast partition of size at most: (a) $d(A)$ if G is $2-$connected; (b) $2 \times d(A)$ if G is $1-$connected.*

Proof: We first analyze the penalty algorithm for two simple $1-$connected and $2-$connected graphs, namely, trees and cycles. Later we use these cases for analyzing general graphs.

Analysis for trees: In the simplest case, let the adversary, w, be a leaf node in a tree T and propagate bogus messages about l genuine identities $v_1, \ldots v_l$ in T. Let $g(v_i)$ be the genuine public keys advertised and $g'(v_i)$ be the fake public keys generated by w. Let P_i represent the path in T from w to the identity v_i. Let r be the last common vertex along all these paths P_i (beyond r the paths diverge, as illustrated in Figure 5.4(a)). Let T_r refer to the sub-tree to the right, rooted at r and $T'(r)$ be the remaining sub-tree that contains w (not containing r). The node r will maintain a penalty of l for all vertices in the path from r to w (excluding r). Hence the value of $s(v_i, g'(v_i))$ is l. However, since the paths diverge at r, the values of $s(v_i, g(v_i)) < l$ for all i. Hence, r will choose the genuine identities v_i over the bogus identities propagated by w. When r filters out the bogus messages from w and broadcasts the chosen identity, the nodes in T_r can reliably broadcast between themselves. On the other hand, any node in $T'(r)$ will associate a value of l for both $s(v, g(v_i))$ and $s(v, g'(v_i))$ since all these nodes (except w) will maintain a penalty of l with both w and r. Hence, these nodes cannot reliably communicate with nodes in $T(r)$ but can broadcast within themselves. Therefore, a leaf node w can create a broadcast partition of size at most 2 in T.

Extending the argument to an adversary w of degree d in a tree T. $T - \{w\}$ has d disjoint sub-trees. Clearly, to cause maximum damage, w should not propagate any good messages across these disjoint sub-trees. Hence w acts as a leaf node in each sub-tree and can thereby create a broadcast partition of at most size $2d$.

Analysis for cycles: Consider the cycle illustrated in Figure 5.4(b) where node w has a left neighbor u_1 and right neighbor v_1. w clearly should not propagate any message from u_1 to v_1 and vice-versa. We call an identity v to be *affected* if w propagates a bogus message about v. Also, we refer to the orientation of the cycle beginning from u_1 as the left portion of the cycle and the opposite

orientation beginning from v_1 to be the right portion of the cycle. To maximize the damage, w should propagate bogus messages about the left portion through v_1 and about the right portion to u_1 (since this minimizes the chance of intermediary nodes filtering w's messages) (refer Figure 5.4(c)). Also, it is to w's disadvantage to propagate two bogus messages about the same identity to both u_1 and w_1.

Let w propagate k_1 bogus messages to u_1 and k_2 messages to u_2. Let x and y be the first unaffected nodes in the left and right portion of the cycle (refer Figure 5.4(c)). Let $g(v)$ be the genuine public key of identity v and $g'(v)$, the fake public key generated by w.

Consider a vertex t in the path u_1x. For every vertex z in the path v_1y, the penalty values of $s((z, g(z))$ and $s(z, g'(z))$ that t maintains are both equal to k_1. Hence t cannot differentiate between the two keyed identities and hence cannot communicate with z. However, for any vertex z in the path u_1x, the penalty value of $s(z, g(z))$is smaller than $s(z, g'(z))$. Hence all nodes within u_1x can reliably communicate within themselves. All nodes in the path xy can reliably communicate their public keys to all the nodes since they are unaffected. In the limiting case, x and y are the same node. Hence, in a cycle, a single adversary w creates a broadcast partition of size 2, namely u_1x and v_1x (with $x = y$).

2-connected graphs: Let G be a $2-$connected graph with a single adversary w of degree $d(w)$. Using Corollary 1, if H is a $2-$connected sub-graph of G comprising only of good nodes, we can merge all these vertices to create a single vertex for H. Let G' be the graph obtained after shrinking all $2-$connected components H to single vertices in G. Clearly $G' - \{w\}$ is a tree since it contains no $2-$connected subgraphs. Also, since G' is $2-$connected, w connects to all the leaf nodes of G' (except in the trivial case when $G - \{w\}$ is 2-connected, G' contains only 2 vertices). We can

extend the cycle argument to show that w can create a broadcast partition of size $d_{G'}(w)$ in G' where $d_{G'}(w)$ is the degree of w in G'. In the case when $d_{G'}(w)$ is even, we can identify $d_{G'}(w)/2$ non-vertex overlapping cycles (except w) and apply the cycle argument to each in isolation. When $d_{G'}(w)$ is odd, we handle a special case where two cycles have an overlapping path fragment (where we use the lower bound argument for $2-$connected graphs to show a broadcast partition of size 3 for overlapping cycles). In the limiting case, we have $d_{G'}(w) = d(w)$ as illustrated earlier in the lower-bound analysis in Figure 5.2(c). Hence, w can create a broadcast partition of size $d(w)$ in G.

1-connected graphs: Let G be a $1-$connected graph with an adversary w of degree $d(w)$. Using corollary 1, if we collapse all $2-$ connected sub-graphs in G to single vertices, the remaining graph G' resembles a tree with cycles involving only the adversarial node w. Let $d_1(w)$ represent the number of neighbors of w in the 2-connected portion of G' (involving w) and let $d_2(w)$ be the remaining set of vertices. Using the previous case, w can create a broadcast partition of size at most $d_1(w)$ in the $2-$connected portion and a broadcast partition of at most $2 \times d_2(w)$ in the remaining portion of the graph (tree case). Hence, the maximum size of a broadcast partition is $d_1(w) + 2 \times d_2(w)$ which is upper-bounded by $2 \times d(w)$ (given $d_1(w) + d_2(w) \leq d(w)$). $\square$

5.3 Power-law random graphs: Proof of Theorem 5

Finally, we provide a proof for Theorem 5 described in Section 5.1 where we show that given a power-law random graph (PLRG) $G(n, \alpha)$ on n nodes with parameter α ($2 < \alpha < 3$), the cumulative damage that a single adversary can cause is bounded by $O(n^{1/\alpha} \times (\log n)^{(5-\alpha)/(3-\alpha)})$ with high probability. We prove two results on power-law random graphs to show this result.

Lemma 12. *Every PLRG $G(n, \alpha)$ for large values of n has a 3-connected subgraph H with $O(n/((\log n)^{\frac{\alpha-1}{3-\alpha}}))$ vertices with high probability.*

Proof: Let G_λ represent the subgraph of G induced by all vertices with a degree at least $\lambda = (\log n + 3 \log\log n)^{1/(3-\alpha)}$ in G. The number of vertices in G_λ is $O(n/\lambda^{\alpha-1})$. Given that in a PLRG, the probability of an edge between two nodes is proportional to the degrees of the two nodes, we estimate the expected degree of every node in G_λ to be at least $d_m = \lambda^{3-\alpha} = \log n + 3 \log\log n$.

Now, we construct a subgraph H from G_λ as follows. For each edge $e = (u, v)$ in G_λ perform the follow step to generate a sub-graph H: Let $d(u)$ and $d(v)$ denote the degrees of u and v in G_λ. Retain the edge e in H with probability $p(u, v) = d_m^2/(d(u) \times d(v))$ provided $p(u, v) < 1$ (otherwise set $p(u, v) = 1$).

In essence, every edge in H occurs with the same probability and one can show that H is Erdos-Renyi graph where each node has an expected degree $d_m = \lambda^{(3-\alpha)}$.

Bollobas proved the following result on Erdos-Renyi random graphs:

[Bollobas85]: *Consider a Erdos-Renyi random graph G on n nodes where every edge is present with a probability p (*i.e.*, average degree of a node is np). G is m-vertex connected with high probability if: $np > \log n + (m-1)\log\log n$.*

Using Bollobas's result [27] on Erdos-Renyi graphs, we show H is $3-$connected with high probability. Hence, the subgraph G_λ with $O(n/((\log n)^{\frac{\alpha-1}{3-\alpha}}))$ vertices is $3-$connected with high probability.

□

Lemma 13. *Given a PLRG $G(n, \alpha)$ and a random vertex v with degree d, the number of vertices that get disconnected from the largest component in $G - \{v\}$ is bounded by $d(\log n)^{(5-\alpha)/(3-\alpha)}$*

with high probability.

Proof: This follows directly by applying Lemma 11 and a conductance result by Gkantsidis*et al.* [56] in power-law random graphs. Let $T(v)$ be the set of nodes that are separated from the largest component $L(v)$ in $G - \{v\}$. Consider a unit flow to be routed between every pair of nodes in G. Any flow from $T(v)$ to $L(v)$ has to be routed via v. Given that all links have unit capacity, Gkantsidis*et al.* show that there exists a way to route the demand such that all links have a flow of at most $O(n \log^2 n)$. Hence, we get the bound that $|T(v)| \times |L(v)|$ is $O(dn \log^2 n)$. From Lemma 11, $L(v)$ contains a 2-connected graph with at least $O(n/((\log n)^{\frac{\alpha-1}{3-\alpha}}))$ vertices. Combining the two results, we obtain the required bound. $\square$

To quantify the cumulative damage an adversary A can cause, let A have a maximum degree in G of $n^{1/\alpha}$. The number of vertices solely reliant on A is bounded by $n^{1/\alpha} \times (\log n)^{(5-\alpha)/(3-\alpha)}$. Given that there exists a sub-graph H which is $3-$connected, A cannot affect any node within this subgraph. Also, all these nodes can reliably broadcast all their messages within H. Additionally, every vertex v has one or more (indirect) neighbors within H. For every identity v that A propagates a spurious message, the penalty value of A increments by 1 and so does the penalty value of the indirect neighbors of v in H. If A targets specific identities such that the indirect neighbors of these identities are distributed among different vertices in H, then A obtains the maximum penalty value and hence is always ignored. To prevent this, A can at most target identities connected to a specific vertex u in H such that the penalties of u and A from the perspective of other nodes is the same. If A targets any additional identity which has a different indirect neighbor in H, then A's penalty overshoots u. Hence, to maximize the cumulative damage, A should target only those identities that solely rely on either A or u or both to connect to H. Using Lemma 12, we can bound this number

by $4 \times n^{1/\alpha} \times (\log n)^{(5-\alpha)/(3-\alpha)}$. $\square$

5.4 Summary and Implications

In this chapter, we proved two important results on the reliable communication in sparse networks for the specific case of a single adversary:

1. For the case of a single adversary in a sparse network, penalty-based filtering with the BROADCAST algorithm described in Chapter 4 is the optimal defense strategy to limit the damage of the adversary.

2. For the specific case of power-law random graphs, the cumulative damage that a single adversary can cause is much smaller than the size of the network.

The first result establishes the rationale behind using the penalty-based filtering algorithm as a defense mechanism in the BGP case in Chapter 3. The power-law random graph result is particularly important for Internet-like topologies where it shows that a single adversarial node will not be able to cause much damage. Specifically, in the case of Internet routing, using this result we can show that the number of incorrect advertisements that a single malicious router can propagate without being detected is bounded. This limits the number of routes that such a malicious router can potentially hijack.

In the next chapter, we describe how the reliable communication theory described in the past two chapters (Chapter 4 and Chapter 5) can be used in practice to secure a variety of different routing protocols.

Chapter 6

Generalizing to secure routing protocols

“ Every generalization is dangerous, especially this one. ”

– Mark Twain, American writer

In this chapter, we describe the design of a reliable communication toolkit that provides a basic set of primitives that can be integrated to provide decentralized security to other routing protocols beyond BGP. The reliable communication toolkit implements the basic theoretical techniques described previously in Chapters 4 and 5 to achieve reliable communication in an unknown network. The primitives exported by the toolkit are independent of the routing protocol. We illustrate the applicability and the generality of the toolkit by integrating it with three standard types of routing protocols, namely: link-state, path-vector and distance-vector routing. Based on a detailed performance evaluation of the toolkit, we show that decentralized security is *feasible, practical and not expensive* under the assumption that the number of adversarial nodes is small.

The rest of the chapter is organized as follows. In Section 6.1, we begin by providing a summary of

our key theoretical results on the reliable communication problem as described earlier in Chapters 4 and 5. In Section 6.2, we describe the basic set of primitives which forms the core set of building blocks that implement the reliable communication algorithms. Next, in Section 6.3, we describe the system implementation aspects of the reliable communication toolkit. In Section 6.4, we consider three basic routing protocols (link-state, distance-vector and path-vector) and illustrate how the toolkit can be integrated with these three routing protocols to secure them in a decentralized manner. In Section 6.5, we describe a detailed evaluation of the reliable communication toolkit. Finally, in Section 6.6, we summarize the key take-away messages from this chapter.

6.1 Reliable communication: summary of key results

Based on our prior results on reliable communication discussed in Chapters 4 and 5, we establish two basic constraints for decentralized security: (a) *fixed-identity criterion*; (b) *graph-connectivity constraint*. The fixed-identity criterion states that every node in the network should have a *unique* identity that it *cannot fake* to its neighbors.[1] While this constraint may seem stringent, existing inter-domain and intra-domain routing use some form of fixed-identities. However, one requires certain basic changes to these systems to strictly enforce the fixed-identity; currently, the criterion can easily be violated. Mobile ad-hoc networks and dynamic P2P networks are two examples of networks that do *not* satisfy this criterion.

The *graph-connectivity constraint* states that one can achieve reliable communication if and only if the minimum vertex connectivity (number of disjoint paths between any pair of nodes) of the underlying network is at least $2k + 1$ given a bound k on the number of adversaries. If this con-

[1] An adversarial node cannot lie about its own identity to its neighbors.

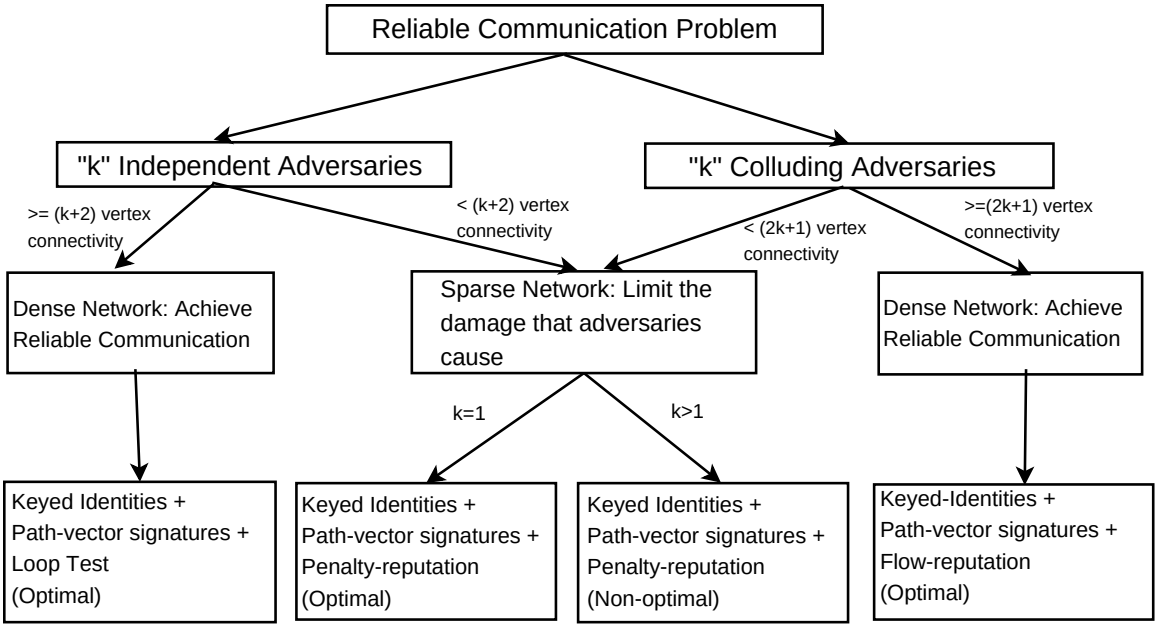


Figure 6.1: Summary of key results on reliable communication. In this figure, k represents the number of adversaries.

straint is not met, we show that it is impossible to achieve all-pair reliable communication in such networks which in turn implies that ”perfect” decentralized security is not possible in such a network. However, for such networks, we use a penalty-based mechanism to restrict the damage that potential adversaries may cause. In reality, given that the vertex connectivity of many existing networks is small (typically < 10), perfect decentralized security is feasible only in the face of a few adversaries.

Figure 6.1 presents a complete summary of our theoretical results on the reliable communication problem in unknown networks and the basic building blocks essential for achieving these results. The following results are important to note:

1. In the face of k adversaries, we require a minimum level of vertex connectivity[2] to achieve

[2]The vertex connectivity of a graph is the number of vertex disjoint paths between any pair of nodes.

reliable communication. A network that does meet this connectivity requirement is defined as a *dense network*; a networks that does not meet this requirement is a *sparse network*.

2. In sparse networks, where this minimum vertex-connectivity constraint is not met, our goal is to limit the damage that adversarial nodes may cause. In a dense network, our goal is to achieve reliable communication.
3. To handle k colluding adversaries, a minimum vertex-connectivity of $2k + 1$ is a necessary and sufficient constraint. However, to handle k independent adversaries, one requires only a minimum of $(k + 2)$ vertex connectivity.
4. Keyed-identities and path-vector signatures [3] are two fundamental building blocks that we use extensively in combination with other techniques achieve reliable communication in different types of networks.
5. In sparse networks, keyed-identities and path-vector signatures along with *penalty-based filtering* is the optimal strategy in the face of a single adversary. However for $k > 1$, the optimal strategy is still an open research question and we apply the same technique that we developed for the single adversary case.

We will now elaborate on the basic building blocks that enable us to achieve reliable communication.

6.2 Reliable communication primitives

In this section, we first elaborate on the basic building blocks necessary to achieve reliable communication. Then, we review the topological structure of several existing networks and comment on

[3] The path-vector signature construction is independent of the routing protocol deployed and should not be confused with path-vector routing. We use this construction to secure distance-vector, link-state and path-vector routing.

the level of security achievable in today's networks. While our basic theoretical techniques assume a specific bound on the number of adversaries (parameter k), we describe a *joint reputation metric* (that combines two different reputation mechanisms) that is independent of the value of k.

6.2.1 Primitives

We use five basic primitives to address the reliable communication problem:

Primitive #1: Keyed-identity: The primary building block in achieving reliable communication is the ability to *differentiate* a message from a good node and a spurious message. To do this, we associate a *keyed-identity (KId)* with every node. A node with a fixed-identity X has its $KId = (X, g(X))$ where $g(X)$ is the claimed public key of X. For every message m transmitted by X, it appends its keyed identity and signs the message with its private key. An adversary generating a spurious message impersonating node X is forced to generate a new KId with a different public-key claim for X. Hence, a truthful message from a good node and a spurious message will have two different keyed identities.

Primitive #2: Path-vector signatures: A path-vector message, (m, p), has two parameters: the message m and the path of identities traversed by the message m. A path-vector signature contains the keyed-identities of the nodes in the path along with their signatures and thereby establishes a transitive trust between the various nodes along the path traversed by this message similar to the web of trust [101] concept used in Pretty Good Privacy (PGP). The property we require is that an adversary should not be able to modify a path-vector message or append (or remove) an identity to the path without modifying the keyed identities of the nodes along the path. Therefore, any spurious message from an adversary is forced to appear from a different keyed-identity path. A

simple signature construction is described in [119].

Primitive #3: Flow reputation: Given two conflicting keyed-identities $(X, g(X))$ and $(X, g'(X))$ about the same identity, X, *flow reputation* is one metric used in dense networks to establish the trust-worthiness of each keyed-identity. From a node Y's perspective, the flow reputation of a keyed identity $(X, g(X)$ is the number of identity disjoint paths that Y has discovered to $(X, g(X))$. In [119], we prove that if the flow reputation of $(X, g(X))$ is greater than the number of adversaries, then $g(X)$ is the genuine public key of X.

Primitive #4: Penalty-based reputation: Penalty-based reputation is an alternative metric used in sparse networks for measuring the trustworthiness of a keyed identity in the face of a conflict. The *penalty* of an identity u is a measure of the potential number of bogus announcements that u might have propagated. Whenever we observe two conflicting keyed-identities $(v, g_1(v))$ and $(v, g_2(v))$ for an identity v, then any node along the common path to either keyed-identity could have propagated a lie; for each such conflict, all these common nodes are penalized equally. From a node x's perspective, given a single path to a keyed identity $(y, g(y))$, the reputation of $(y, g(y))$ is inversely proportional to the *maximum penalty* of an identity u that appears in all paths from x to $(y, g(y))$. We denote this maximum penalty as $max(y, g(y))$. An identity y that has m different keyed-identity claims $(y, g_i(y))\forall 1 \leq i \leq m$ (all of which have a single disjoint path), has a normalized reputation of:

$$\frac{1/max(y, g_i(y))}{\sum_{i=1}^{m} 1/max(y, g_i(y))}$$

The penalty-based filtering strategy of choosing the route to an identity with the highest reputation is the optimal strategy in the face of a single adversary [119].

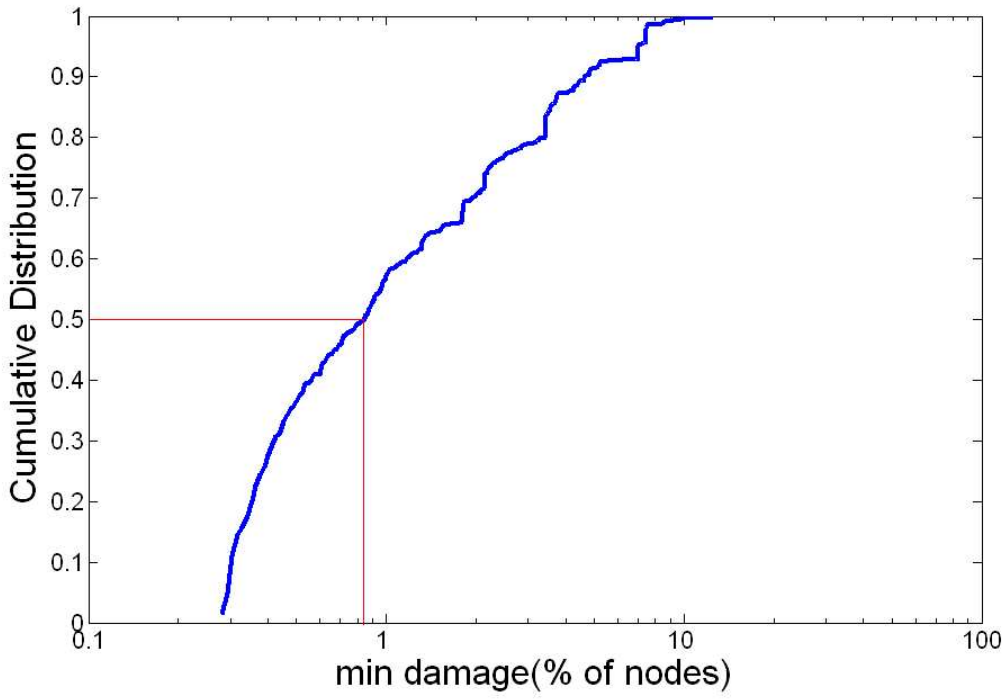


Figure 6.2: Effectiveness of penalty-based filtering: The fraction of nodes that are vulnerable to a single randomly placed adversary after. This result is computed for the AS topology collected in Jan 2005 based on Routeviews and RIPE data.

Primitive #5: Loop Test: Loop-test is used to defend against independent adversaries where a node source-routes a packet (with a nonce) in a loop and tests whether it successfully traverses the loop. If the loop consists of fake edges generated by independent adversaries, the loop-test will fail. However, colluding adversaries can defeat this mechanism by tunneling the loop-test. This test is essential to show that a vertex-connectivity of $k + 2$ is sufficient to deal with k independent adversaries.

6.2.2 Achievable security in existing networks

Based on theoretical results summarized in Section 6.1, we will now describe the connectivity properties of different networks and relate it to the corresponding level of security achievable for these networks.

Internet-AS topology: The Internet AS-topology is a sparse network that is $1-$vertex connected. Therefore, one cannot achieve reliable communication even in the face of a single adversary in such a network. Today, a single randomly-placed router can hijack nearly one-third of Internet routes by propagating bogus information [84, 120]. The best one can do is to limit the damage that an adversarial node can cause using the penalty-based filtering. Figure 6.2 shows the effectiveness of this mechanism on a specific AS-topology gathered from BGP routing data. We observe that a single randomly placed adversary, in the median case, can hijack less than 1% of nodes in the topology. While the overall topology may not satisfy the connectivity requirements, portions of the topology exhibit high vertex-connectivity. Figure 6.3 shows the connectivity at different levels within the AS hierarchy. The set of Tier-1 and important Tier-2 ASs form a 19 vertex connected graph. Hence, it is possible to defend against 9 colluding adversaries within this sub-graph *i.e.*, using the flow-reputation mechanism. Similarly, one can provide different levels of security protection for various portions of the Internet topology. In particular, a large fraction of transit networks in the Internet form a $3-$connected subgraph in which we can defend against a single adversary. Additionally, multi-homing improves the connectivity of the underlying topology which in turns improves the achievable security properties of the Internet. An AS that is multi-homed to two upstream providers is not susceptible to attack from a single adversary as long as the adversary is not present in one of the upstream paths to the core of the Internet.

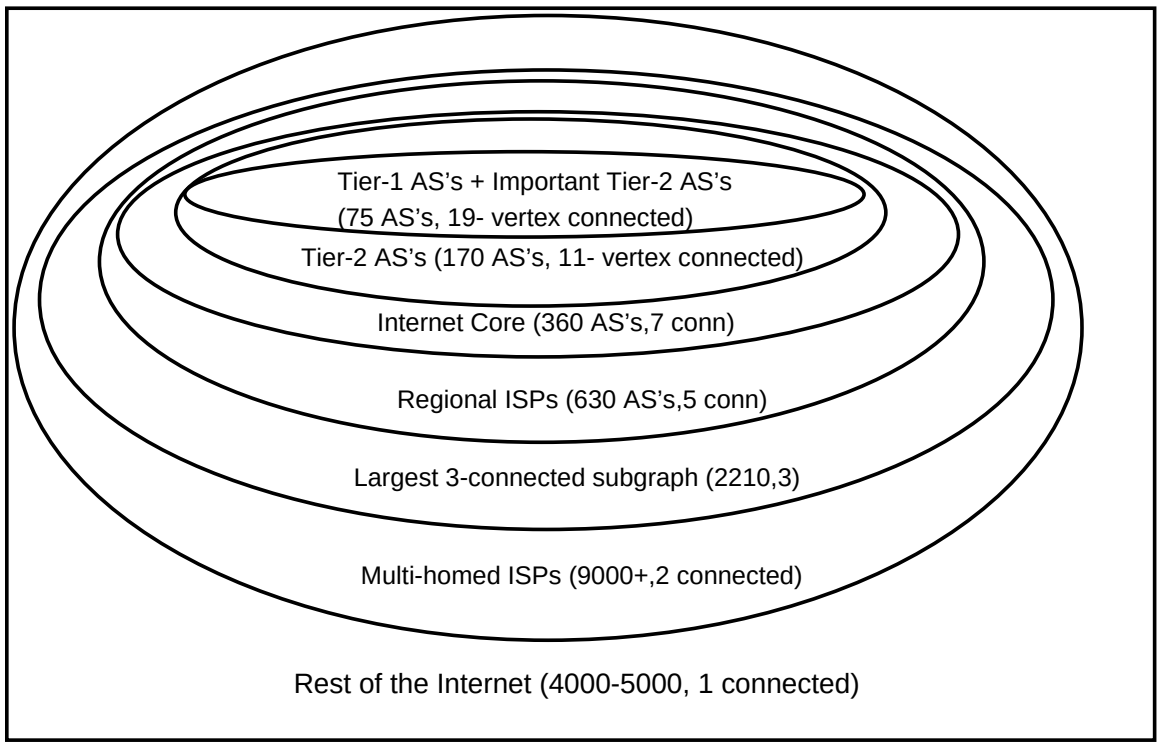


Figure 6.3: Internet AS-topology: Different levels of protection.

	k=1	k=2	k=3	k=4
AS1221 (Telestra)	210	93	51	23
AS1239 (Sprint)	622	275	195	60
AS7018 (AT&T)	568	238	160	58
AS1755 (EBone)	101	49	23	11
AS2914 (Verio)	689	249	172	62
AS3967 (Exodus)	231	53	22	13

Table 6.1: Level of protection in Intra-domain routing. Each entry represents the largest $(2k+1)$ vertex connected subgraph within a domain for a given value of k.

Intra-domain networks: The router and the POP level topologies of individual domains also exhibit different levels of connectivity for different sub-graphs. Table 1 shows the size of the largest subgraph of an intra-domain network which is $2k + 1$-vertex connected for different values of $k = 1, 2, 3, 4$. We make two important observations. First, the sub-graphs that exhibit high connectivity properties are POPs in well-populated cities forming the backbone of the network providing better security to the backbone in the face of a few adversaries. Second, a system administrator can easily add a few more intra-domain routing sessions to explicitly construct a $3-$connected network to defend against a single adversary. For the Sprint and AT&T networks,this would involve adding at most $1 - 2$ additional sessions per router (mostly to geographically-proximate routers). In essence, a system administrator can configure the connectivity to improve the level of security.

Structured P2P networks: Recent research efforts have argued for using structured P2P networks [105, 117, 109, 131] for a variety of services including essential services like the Domain Name System [103, 58, 42], worm-defense systems [31, 130]. Existing security solutions for such networks use a PKI with a certification authority [32]. Such networks can potentially be associated with fixed identifiers. Reliable communication can be useful for routing information reliably in the face of adversaries between the various nodes in the system. Several existing P2P network technologies [105, 117, 109, 131] use different types of $d-$regular graphs (all nodes have degree d) for different values of d. In such networks, we can achieve reliable communication in the face of $d/2 - 1$ adversaries.

6.2.3 A joint reputation metric

Our theoretical results assume a specific bound k on the number of adversaries and provide provable security guarantees assuming such a bound exists and is known in advance. However, in real world networks, it may not always be possible to fix a specific value of k and in practice, one may require different values of k for different sub-graphs of the network as observed for the Internet AS-topology and intra-domain networks. In order to accommodate this variability in different values of k, we define a *reputation metric* which combines the salient aspects of the flow based metric and the penalty-based metric. For a specific keyed-identity $(y, g(y))$, a node x sets the reputation, $rep_x(y, g(y)$, as:

1. If x has two or more identity disjoint paths to $(y, g(y))$, then $rep_x(y, g(y)) =$ number of disjoint paths to $(y, g(y))$.
2. If $(y, g(y))$ has just one disjoint path and no conflicting identities. then $rep_x(y, g(y) = 1$.
3. If $(y, g(y))$ has just one disjoint path and has conflicting identities. then $rep_x(y, g(y) =$ penalty-reputation of $(y, g(y))$.

This joint reputation metric satisfies two properties. First, if the reputation of a keyed-identity is more than a known bound on the number of adversaries, then that keyed-identity claim is genuine. Second, for a given identity with a conflict, the keyed-identity with a higher reputation is more trust-worthy than the one with a lower reputation. The mechanism of choosing routes based on the highest reputation is the optimal strategy to defend against a single adversary. On the other hand, in the face of multiple adversaries (where the number of adversaries is unknown), one can use the reputation as a guiding factor and choose routes probabilistically. However, providing any form of

security guarantees for probabilistic reputation-based route selection is hard.

6.3 The reliable communication toolkit

In this section, we will describe the reliable communication toolkit and discuss the salient design issues of our implementation.

6.3.1 Interface

The reliable communication toolkit provides routing protocols a set of primitives to achieve secure routing. It is useful to think of the toolkit as a layer that is used by the routing protocols to communicate securely. This modular design enables it to be easily integrated with the routing protocols. The toolkit exposes the following functions (which implement the primitives) to each node that implements the toolkit:

Keyed-Identity functions: $gen\_keyed\_identity()$ and $sign\_message(msg)$ are the basic functions for generating a keyed identity and signing messages (typically route updates) using the private key. All public key operations are based on 1024-bit RSA keys.

Path-vector signature functions: The three basic functions associated with path-vector signatures are: $initiate(path\_vector)$, $verify(path\_vector)$, $update(path\_vector)$. The $initiate()$ function initiates reliable communication of a value and generates the path-vector containing the value and the corresponding signature. The $verify()$ function verifies the authenticity of a path vector originating at another node. This makes use of public key signatures and chained signatures for verification. Given a path vector, if the $path\_vector$ cannot be authenticated, it is discarded. The

Variable	Description
$nodes$	path vectors to keyed identities
$reliable\_nodes$	path vectors to reliable keyed identities
$edges$	edges discovered
id_count	vector of counts for the IDs seen
$keyed\_id\_count$	vector of counts for the keyedIDs seen

Table 6.2: The basic state maintained by the toolkit

$update()$ function updates an authenticated path vector along with the identity of the node receiving the vector.

Reputation functions: The $get\_keyed\_identities(ID)$ outputs the different public-key claims that have been learnt for a given identity. The $get\_reputation(keyedID)$ implements the joint reputation metric discussed in Section 6.2.3. The $flow\_reputation()$ and $penalty\_reputation()$ perform the flow and penalty reputation computations. Additionally, the $looptest(keyedIDloop)$ performs the loop-test functionality on a keyed-identity loop.

6.3.2 Implementation

Each node must run the reliable communication process to be able to utilize routing protocols securely. The toolkit is a single-threaded event-driven program. The toolkit was written in C and C++ comprising about 6000 lines of non-comment code. The implementation includes the functions that implement the primitives, the identity-based scheduler (Section 6.3.3) and the secure routing protocols (Section 6.4).

The state maintained by a node is shown in Table 6.2. A node, on booting up, first goes into a bootstrap phase in which it initiates the reliable communication, and then into a propagation phase

in which it learns the nodes. $nodes$ is a table of path-vectors to all keyed-identities, $edges$ is a compact representation of all the edges in different paths and $reliable\_nodes$ refers to the set of nodes discovered reliably ($flow\_reputation() > k$). id_count and $keyed\_id\_count$ keep track of the number of messages that are received by the node containing an ID or a keyedID respectively and is used by the scheduler.

The toolkit can be decomposed into three modules.The *network module* handles the sending and receiving of the different types of packets, including the marshaling and unmarshaling. The *packet processor module* upon receipt of each path-vector packet: (a) verifies the signature; (b) adds the path vector to the source $nodes$; (c) schedules an authenticated path-vector packet for dispatch; and (d) at regular intervals, performs the flow computation to estimate the reputation of identities and makes the necessary updates to the $reliable\_nodes$ data structure. The *scheduler module* is used to ensure that a malicious node does not prevent the remaining nodes from getting their packets across by flooding the network. We elaborate on the scheduler in Section 6.3.3.

Bootstrap phase: Consider a node A that joins the network. A needs to build a keyed identity graph over the set of nodes that it has discovered reliably. It reads its neighbor set from a configuration file and connects to each of these nodes. A creates a public-private key pair (K_A^{pub}, K_A^{priv}). (A, K_A^{pub}), which we shorten to (A, K_A) is the keyed identity of A. It sends each of its neighbors an $send\_keyed\_id(KId)$ message containing its public key. Each neighbor sends an $send\_keyed\_id(KId)$ message containing its public key. Node A then broadcasts a $initiate(path\_vector)$ message to each of its neighbors. The path-vector message that A sends to a neighbor B is of the form $(m(A), [(A, K_A), (B, K_B)], sign_A)$ where $m(A)$ is the data that A wishes to broadcast and $sign_A$ is A's signature on $(m(A), [(A, K_A), (B, K_B)])$ using its private

key.

Propagation phase: Propagation of path vector messages is triggered off by two events - by the arrival of a path vector message and by the arrival of a new node in a node's neighbor set (specifically when a node receives an $send\_keyed\_id(KId)$ message from a neighbor with a new keyed identity). In the former case, the message is simply queued at the scheduler to wait for its turn to be propagated. The latter case occurs whenever a new node or a new link comes up. In this case, the node examines the path vectors in the $nodes$ table and $reliable\_nodes$ table. It then queues these messages at the scheduler for dispatching to its new neighbor. A node that receives a path vector message, verifies the chained signatures in the path vector message using $verify(path\_vector)$. If the message is authentic, it adds newly learnt edges or vertices to the keyed-identity graph G_A and uses $get_reputation$ to assign reputation to the keyed ids *e.g.* for computing the flow reputation it uses $flow(reputation(keyedId) > k$. The updated message is then queued to be broadcast to the remaining neighbors. When the scheduler notifies the packet processor of a message to be broadcast, it calls $update(path\_vector)$ on the message and forwards the message.

6.3.3 Design Issues

Path Suppression

One of the fundamental problems in propagating path-vector messages is the exponentially large number of distinct paths and hence, unique path-vector messages. The worst case convergence time for a path-vector routing protocol is exponential in the number of nodes [75]. To address this combinatorial explosion, we use the idea of *path suppression* where every node *explicitly suppresses a*

path-vector routing message in the event where it does not contain any additional information compared to previous messages. More specifically, if a node A has already propagated m path-vector messages with paths $P_1, \ldots P_m$ to its neighbor B, then A transmits a new path-vector message with path P_{m+1} to B only if P_{m+1} contains a new edge or a new vertex not present in $P_1, \ldots P_m$. In the absence of any adversaries, the number of path-vector messages propagated on any link is bounded by the number of edges in the graph.

The Scheduler

We show in our prior work [119] that simple Fair Queuing or FIFO scheduling is inadequate to protect against malicious flooding attacks in bandwidth-constrained networks. To mitigate such an attack, we make use of identity-based scheduling. The scheduler has two vectors of counters - id_count the *identity priority* counter and $keyed\_id\_count$ the *keyed-identity priority* counter. For a given identity, id_count counts the number of path vector messages received that contain this identity. The Identity priority of a path vector message $IP(M)$ is the *maximum value of the identity priority* counter over each identity in the path. $keyed\_id\_count$ counts the number of path vector received messages with a given keyed identity. The Keyed-identity priority of a path vector message $KIP(M)$ is the *maximum of the keyed-identity counter* over each keyed identity in the path. The scheduler gives highest priority to messages with low $IP(M)$, breaking ties favoring low $KIP(M)$, and finally using FIFO. We will later demonstrate in Section 6.5 that identity-based scheduling provides fast convergence even in the face a flood attack from an adversarial node.

The scheduler is implemented as a simple priority queue. For this scheduling discipline, the addition of a new message to the queue does not alter the priority of the remaining messaged. The

priority queue allows both enqueueing and removing the highest priority message to be performed in $O(\log n)$ time. When the buffer fills up, messages with the lowest priority are dropped (this is a simple operation that involves dequeuing messages from the end of the queue).

Maintaining State

The third important design issue that we faced while implementing the toolkit was the maintenance of state associated with reliable communication *i.e.*, how does each node maintain its keyed identity graph as the underlying topology continues to evolve with time (node arrivals/failures)? The two options that we need to consider are hard state and soft state. In a soft state approach, each node in the keyed identity graph has associated with it two timers. One timer τ_1 is started when the node is discovered and the other τ_2 when the node is identified to be reliable. When τ_1 fires, all path vectors associated with the node are discarded. when τ_2 fires, the path vectors are discarded and the node is no longer considered reliable. Nodes must re-initiate reliable communication periodically to keep the state fresh. This periodic operation increases the cost of providing reliable communication. Further, it can be abused by adversaries to flood the network during each repeated broadcast.

The hard state approach simply involves reliable communication whenever a new link comes up. Nodes that are reliably discovered remain so irrespective of any changes in the topology. This is acceptable because once a node has been reliably discovered and the knows the other node's public key, the underlying topology is irrelevant in exchanging data reliably. So a node A that has reliably discovered node B will continue to believe B to be reliable even when B goes down and comes back up provided B has the same keyed identity (B, K_B). This also ensures that reliable communication is not triggered off by transient events such as link failures. If node B changes its keyed identity

to (B, K'_B) however, then reliable communication is again initiated. Node A replaces the former keyed identity of B- (B, K_B) with its new identity (B, K'_B) if it can again discover $k+1$ identity-disjoint paths to (B, K'_B). Further, in a hard state design, the adversary cannot cause much damage after the initial reliable broadcast phase (Any bogus updates propagated by the adversary will be given a very low priority by the scheduler making it hard for the adversary to flood the network). For these reasons, we have used a hard state approach in our system.

Liveness of State

While securing routing protocols, we guarantee that a node cannot claim to have a non-existent link. However, this does not prevent a node from forwarding packets along a path different from the one advertised by it or from not forwarding any packets at all. This problem is a data plane issue while our solution helps to secure the control plane alone. Thus, the liveness guarantees provided in the control plane are limited and we need to use data plane techniques like [120, 81] to address liveness. Deployment of secure routing does not prevent malicious nodes from affecting the traffic.

This issue also ties in with another aspect of routing protocols - refreshing the routes. Refreshing of the routes ensures that nodes have the most up-to-date information required to forward packets. However in the presence of data plane techniques for detecting liveness, periodic refresh of routes can be done away with. This is desirable from the point of reducing the overhead of maintaining up-to-date routes.

6.3.4 Optimizations

Now, we describe some optimizations that further bring down the cost of protocol operations.

Using neighbors' information: When a node is trying to discover the graph, it can make use of the path vector computations performed by its neighbors. If at least $k + 1$ neighbors of node A claim that a node X is reliable, then A can consider X to be reliable. This does not require A to know any of the path vectors containing X. This is particularly beneficial when a new node joins the network - it learns most of the nodes in the network simply from its neighbors. Nodes periodically propagate any new reliably discovered nodes to their neighbors.

Lazy propagation: When a new node joins the network, a neighboring node can lazily propagate path-vector messages. Instead of propagating all the path vector messages it has received to the new node, it sends the graph and its edges alone. It then forwards all the reliable identities learnt. The new node determines the reliable nodes based on the updates from its neighbors. If there exist nodes in its graph that have not been reliably discovered, it then requests each of its neighbors for signed path vector originating from these nodes. These are then used to determine if the nodes are reliable. This lazy propagation of path vector messages reduces the cost of node joins.

Symmetric keys and one-time signatures: After a node has reliably discovered other nodes in the network, it can use the secure channel to exchange symmetric keys or setup a one-time signature exchange. This ensures that the frequent updates are inexpensive. In the latter case, nodes will have to periodically renew their one-time signatures. This optimization is particularly useful for the case of distance-vector routing where nodes only exchange routing information with their neighbors and their neighbors' neighbors.

Path vector reuse: In secure path vector routing, when a node finds that the current path to a destination goes down, it needs a new path vector before it can again route to the destination. This path vector must originate from the destination with a higher sequence number than the last path vector sent out. This increases the time to convergence of the routing protocol. On the other hand, other nodes may know of alternate genuine paths to the destination. These nodes could, potentially, propagate the path vector to the destination so that nodes can repair their broken routes. This requires nodes to be allowed to reuse path vector messages. However, the protocol is now vulnerable to replay attacks in which a nod claims reachability even after the path no longer exists. This violates the monotone property that we would like the routing protocol to guarantee. An alternate take views this attack as equivalent to one in which the malicious node that advertises a path to a destination drops or misroutes packets in the data plane. In the presence of mechanisms to detect data plane malice, we can allow nodes to reuse path vectors to improve performance.

6.4 Secure routing

In this section, we describe how secure versions of three standard routing protocols, namely, link-state, distance-vector and path-vector routing, can be layered on top of the reliable communication toolkit. One *important distinction* between reliable communication and secure routing is in order. The former is a nearly-static protocol to reliably discover the network while the latter is a dynamic process that needs to setup routes in the network. Once reliable communication has been established, a routing protocol can use it as a building block to reliably propagate routing information in the face of adversaries. The routing protocol is independent of the actual mechanism used to establish reliable communication.

In general the routing protocols that we describe make use of the following functions provided by the toolkit: a) the path-vector signature functions for the route updates, and b) the reputation functions to compute the reputation associated with a node. When the routing protocol receives a packet, it evaluates the reputation associated with the nodes in the packet *e.g.*, in a dense graph, it would compute $flow\_reputation()$ associated with the relevant nodes in the packet and accept the packet only if their reputation is $> k$. The protocol then uses the path-vector signature functions to perform updates to the packet. We present the details in sections 6.4.1, 6.4.2, and 6.4.3.

The basic security property that we desire from the protocol is that *no node must be able to propagate routing information that is inconsistent with the state of the network at that time.* More concretely, we require two security properties to be met:

Property 1. *No node can claim a route to a destination that includes non-existent links in the graph.*

Property 2. *A routing protocol has the monotone update property if and only if the cost of the path claimed by a node to a destination (other than itself) is not lower than that of the least-cost path from any of its neighbors.*

We make two observations. First, property 2 arises in the context of distance-vector routing where we require that an adversary should not be able to reduce the cost of a route. However, any node can always increase the cost of a route. Second, two colluding adversaries can always prove the existence of a genuine link between them; such links are considered to be existent links in the underlying graph.

6.4.1 Secure path-vector routing

The reliable communication toolkit is a generalization of the Whisper technique developed in our prior work [120] as a mechanism to secure BGP, a path-vector routing protocol. Therefore, to secure path vector routing, we can directly leverage the chained path-vector signature construct used in reliable communication. Any bogus announcement containing a non-existent link to a genuine node will result in a keyed-identity conflict where we revert to the reputation mechanisms to determine the genuine keyed identity. While applying the toolkit to BGP, two specific issues arise:

Propagating address ownership: BGP routes at the granularity of prefixes as opposed to based on the destination identities (as performed by the toolkit). Hence, to do prefix-level routing, each AS needs to propagate ownership claims for the prefixes they own as part of the initial keyed-identity claims. Here, unfortunately, one cannot blindly use flow-based reputation mechanism to address ownership conflicts. If an adversary raises only a single bogus claim for a specific prefix, it is fundamentally impossible in a decentralized setting to determine the identity of the real-owner of the prefix. This is a fundamental limitation of decentralized security in comparison to a PKI-based approach where the ownership is authenticated by the PKI root. The best one can do is to use a penalty reputation scheme to prevent adversaries from generating several bogus ownership claims. One can show that penalty reputation is the optimal strategy even in the face of several adversaries.

Route aggregation: BGP allows route aggregation which enables a router to aggregate routes to two prefixes. The operational practice of BGP today suggests that route aggregation is either performed by the owner of a prefix or the first upstream provider who allocated the prefix-range to its customers [50]. To support such aggregation, we simply concatenate the two signatures associated

with the individual sub-prefixes. However, if the origin ASs of two prefixes are different, aggregation is explicitly not allowed in reliable communication since the identity of the origin becomes hidden.

6.4.2 Secure link-state routing

Once reliable communication is established, every node is aware of the entire topology and has a trusted channel to every node in the system. To secure link-state routing, we use the trusted channels to establish two-way authentication on each link *i.e.,* a link-cost update on a link (A, B) is genuine only if the cost has been authenticated by both A and B. Unless a two-way authentication is obtained, a link is not considered as part of the topology for routing purposes. Instead of needing two separate update messages for a link cost-change, each node initially obtains a signature from its neighbor and appends both the signatures (including its own) as proof to each update. However, when a link fails, each node will simultaneously propagate the link failure information independently without two-way authentication; the link is treated as a failure as long as one of the messages reaches a node. Each link-state updates is flood reliably. Since trusted channels are pre-established, flooding is the same as traditional link-state flooding along every link coupled with the additional cryptographic operations. Hence, the subsequent cost of propagating link-state updates is the same as traditional link-state flooding except the cryptographic checks. In order to support secure OSPF routing across areas (link-state routing within an area, distance-vector across areas), it is essential to use reliable communication to learn all the nodes across areas but then revert to secure distance-vector routing across areas.

6.4.3 Secure distance-vector routing

Distance vector (DV) routing is hard to secure because of the limited information that is passed around. This makes it easy for a malicious node to claim a fraudulent low-cost path or a non-existent link to a destination node. We describe two simple techniques to mitigate this problem. The first approach is to use the *hash-chain signature* mechanism used in prior work [63], where every origin initiates a routing announcement based on a seed s and a pre-specified one-way hash function $h()$. Every intermediary node incrementally hashes the signature such that a route of cost c has a signature $h^c(s)$ (assuming c is an integer, $h^c()$ is the $h()$ applied c times); given $h()$ is one-way, an adversary cannot reduce the cost. Any node can use the hash-chain signature to check for consistency across different routes [63, 62] – however, to verify authenticity of a single signature, a node needs to use the trusted channel to the origin. An alternative simpler strategy to defend against independent adversaries is to use *lookahead information* from the neighbor's neighbor. In this scheme, whenever a node propagates a DV update to its neighbor, it needs to: (a) sign the DV update (including cost, origin, its identity) using its private key; (b) forward the signature of the neighbor it received the update from. Given that reliable communication is established, the neighbor's neighbor signature can be verified using the pre-established trusted channel. By doing so, one can prevent an independent adversary from propagating bogus announcements since it cannot generate a genuine neighbor's signature. This signature mechanism does not require involving the origin to verify the authenticity of the signature.

6.4.4 Protecting against replay

To have up-to-date routing information, the routing protocol must propagate updates at regular intervals. The routing table entries are also maintained in soft state so that a node that does not receive an update with a time window removes the corresponding entries from its routing table. However, an adversary may replay old update messages to claim the existence of a link that has gone down. This is solved by including with each update a sequence number - the sequence number must be an increasing number. A node checks each update to see if it is a higher sequence number before using it in its routing table. In path vector routing, for a path vector update originating from node A, node A is responsible for using an increasing sequence number and signing it to ensure that it cannot be modified. In the link state case, for a link between A and B, node A gets its sequence number signed by B. This ensures that A cannot replay an old response from B claiming that the link $A - B$ exists. A includes this along with the sequence number in its link state advertisement which it again signs. This ensures that no other node can replay the advertisement. In the distance-vector routing, a node A again shares its sequence number with its neighbors. Every distance-vector update contains the sequence number of both the nodes sharing the link.

6.5 Evaluation

In this section, we evaluate the system performance of the reliable communication toolkit. Our evaluation is targeted at answering three questions: (a) What are the costs of the functions implemented by the toolkit? (refer Sec 6.5.1); (b) How do the system-wide properties (time to converge, message complexity) vary as we scale the system? (refer Sec 6.5.2); (c) What is the cost of integrating

routing protocols with the reliable communication toolkit? (refer Sec 6.5.3). Separating the routing protocols from the toolkit makes sense because reliable communication is invoked at the frequency with which new identities appear in a node's neighbor set and this is infrequent. Routing on the other hand happens whenever link costs change. This separation allows us to consider the one-time and the recurring costs of secure routing in isolation.

Our experiments show that the toolkit can implement the basic operations at rates comparable to a BGP router. The time to complete reliable discovery is around 15 seconds in a 50 node network. The routing protocols layered on the toolkit have convergence times close to the normal routing protocols -the difference being on the order of seconds.

6.5.1 Microbenchmarks

We measure the time taken by the toolkit to perform its basic operations. We also study the behavior of these metrics with increasing path vector length. The setup comprises three machines each running a reliable communication process. Each machine is a quad-3.06 GHz Intel Xeon. The topology assumed by the reliable communication processes is a chain. Each experiment is parametrized by the value of the path vector length. The nodes at either end of the chain generate path vectors of the specified length and propagate it along the chain. Each experiment is run for 2 minutes and repeated 10 times. The path vectors are varied from single-hop vectors to those with a hop length of 20. We use 1024-bit RSA keys in all cases.

Operation	Avg time with 95% CI (μsec)
$initiate()$	4326.53 $\pm$ 30.25
$generate\_key()$	149710.864 $\pm$ 5234
$update()$	3992.28 $\pm$ 4.90
$verify()$ (path length=10)	2579.73 $\pm$ 4.77

Table 6.3: Operations involved in reliable communication.

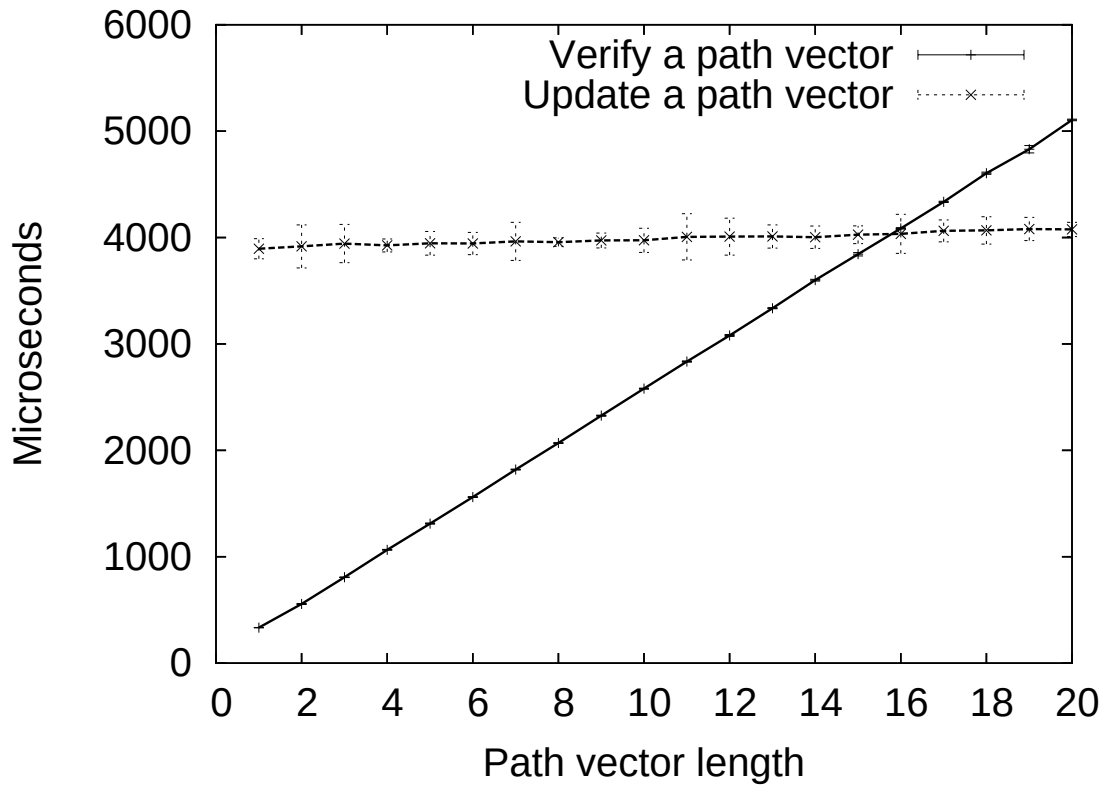


Figure 6.4: Time to verify and update a path vector message as a function of the path vector length

Time taken for unit operations

The processor-intensive operations in the reliable communication toolkit are primarily the cryptographic operations namely: generate a public-key $generate\_key()$, the time taken to initiate a path vector $initiate(path\_vector)$, the time to verify a path vector $verify(path\_vector)$, and the time taken to update a path vector $update(path\_vector)$. Table 6.3 shows the mean-time consumed by each of these operations. The basic path-vector signatures average roughly in the order of a few milliseconds while $generate\_key()$ is a much more expensive operation. The $generate\_key()$ operation is a relatively infrequent operation and is reinvoked only if a node intends to change its public-key. The time complexity of $verify()$ scales linearly with the path-length since each verify operation requires a node to verify the signature corresponding to each hop of the path vector. Figure 4 illustrates the variability of the $verify()$ as a function of the path length. For path lengths up to 6 hops (most BGP AS paths have a path-length less than 6), the verification takes less than $1.5ms$. This implies we can process roughly $40,000$ route updates per minute which is 4 times larger than the maximum update rate observed at a BGP router [13]. The time to update a packet is constant across path vector lengths since it involves appending a single signature to a path vector.

6.5.2 System-wide properties

The next set of experiments is intended to test the system-wide properties including the *time taken by nodes to reliably discover (TOR)* other nodes in the network and the *communication overhead*. Our testbed comprises the 62-node Millennium PSI fast-storage and compute cluster [25]. Each node is a dual 3.0GHz Pentium4 Xeon with 3GB of RAM. We specifically chose *not* to run multiple instances in the same machine since the contention for local resources of different instances within the same

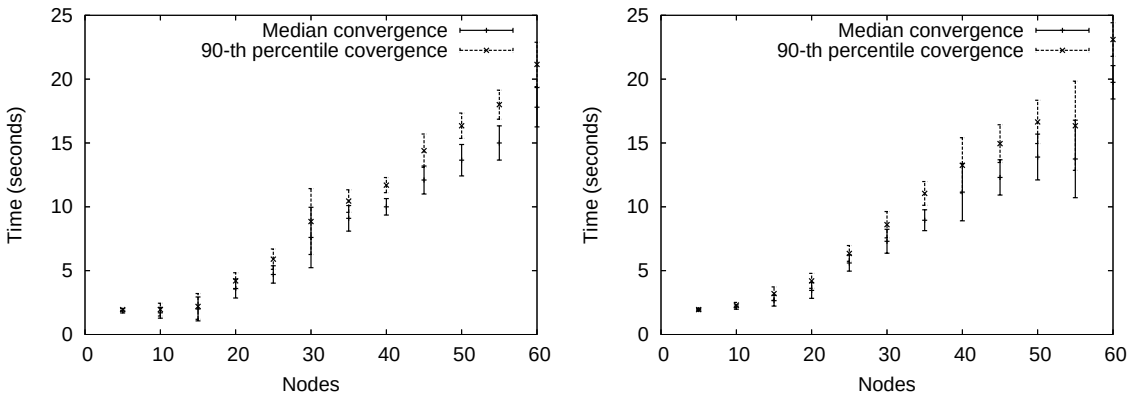


Figure 6.5: a)Time taken for nodes to reliably discover(TOR) other nodes in the network as a function of network size. b) TOR as a function of the network size in the presence of a single malicious node

machine greatly affected the system-wide properties we measured (especially since some of our operations are crypto intensive). Hence, we report system-wide properties only for a maximum system-wide configuration of 60 nodes. Additionally, we chose to run our experiments at time-periods when the resource contention from other processes in this shared cluster was minimal (if not none). We generate random m-connected topologies by constructing a random graph G_d where each node has an edge to d nodes chosen uniformly at random. For a collection of n nodes, by setting $d = \log n + (m-1)\log logn$, we can obtain a graph that is $m-$vertex connected with very high probability [27]. We consider two scenarios for measuring system-wide properties: (a) Simultaneous start-up; (b) Single node arrival.

Simultaneous start

In this setup, all nodes are initiated simultaneously. We consider two scenarios: i) all nodes are well-behaved, and ii) there's a single malicious node that propagates bogus path vector messages every second.

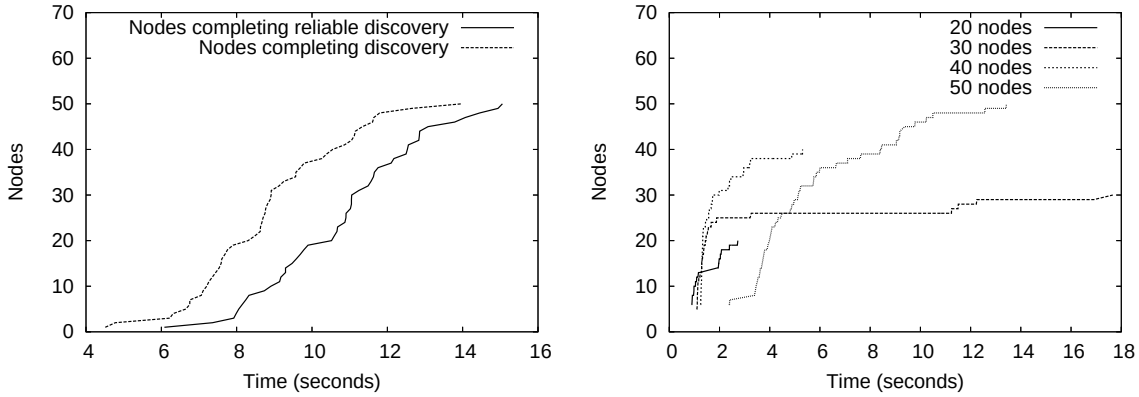


Figure 6.6: a) Time taken by different nodes to discover and reliably discover all other nodes. b) Time taken by a single node to reliably discover all the nodes in the network for different network sizes.

Nodes	Messages	MBytes
60	3249.47	5.66
50	2701.64	4.59
40	2035.65	3.34
30	1554.13	2.64
20	929.1	1.33
10	295.2	0.35
5	80	0.75

Table 6.4: Average number of packets and bytes transferred when all nodes join simultaneously.

Figure 6.5(a) shows the median and 90-th percentile of the time taken to reliable discovery (TOR) as a function of the network size. This value has been averaged over 20 runs. We see that the TOR increases slowly with network size. Even for a 50 node network, the 90-th percentile is under 20 seconds even in the presence of the malicious node (Figure 6.5(b)). In fact interestingly, the time to converge decreases slightly in malicious presence. This happens because the spurious messages injected results in nodes learning about the other genuine nodes more quickly. The other interesting observation regarding the trend is that the median TOR curve flattens out with more nodes suggesting that for a given connectivity the TOR usually depends on the diameter.

Figures 6.6(a) and 6.6(b) give us some more insights into the reliable communication process. Figure 6.6(a) shows that the time interval between a node being discovered and being reliably discovered is of the order of a second. Thus, the overhead of waiting for multiple path vectors and computing identity-disjoint paths is small. Figure 6.6(b) shows that most of the reliable discovery happens during short span of time - the time interval between when the number of nodes discovered is about 5 (only neighbors discovered) to the entire network being reliably discovered.

The overhead of reliable communication in terms of the time taken to reliably discover nodes before routing is much smaller than the the times shown in Figure 6.5. We also measure the communication overhead in terms of the number of bytes sent and received, as shown in Table 6.4. In a 50 node network, nodes send and receive about 4.5MB (2500 messages) over an entire session on average which is an acceptable overhead in most networks. This is comparable to node discovery by flooding which would require 300 messages on the 50 node topology that we have used (a message traversing an edge in each direction). In steady state, the median memory footprint is about 12MB with a maximum footprint of about 20MB.

We make two observations. First, the scenario of nodes joining the network simultaneously is not very realistic though it gives us a worst-case estimate of the cost incurred by reliable communication. Second, the code has not been optimized for performance. The results reported here are more useful as an indication of the performance rather than as an accurate estimate.

Single node arrival

In this scenario, we consider a more plausible scenario of node joins. We have a set of 49 nodes that have reliably discovered one another. A new node now joins the network. We measure the time taken for the old nodes to reliably discover the new node and vice-versa. This is again shown in figure 6.7(a) and 6.7(b). We see that it follows the same trends as the graphs in Figure 6.5 though the discovery happens in about 10 seconds. Communication overhead numbers for this scenario also show that the overhead incurred is similar to the simultaneous join case (5.1MB on the average).

To summarize, we find that with respect to the most interesting system-wide property of TOR, the system shows low convergence delays, even in the presence of bogus traffic, and has an acceptable overhead.

6.5.3 Routing protocol

To evaluate the cost of integrating security into routing protocols, we implemented simple versions of path-vector, distance-vector and link-state routing. These protocols make use of the reliable communication layer to perform the initial key distribution. These protocols can be run in secure and insecure modes. We ran each protocol over a randomly-generated 50 node network in which all

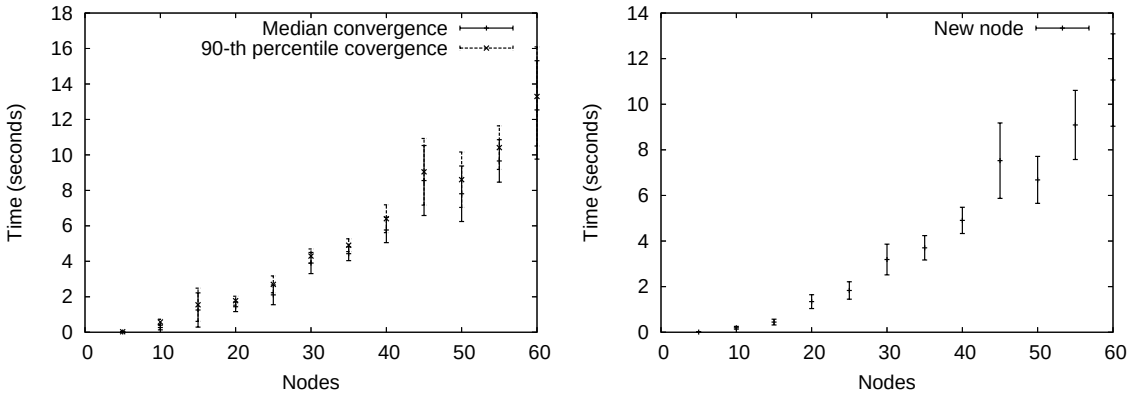


Figure 6.7: a) Time taken for the network to reliably discover a new node for different network sizes. b) Time taken by a new node to reliably discover the network as a function of the network size.

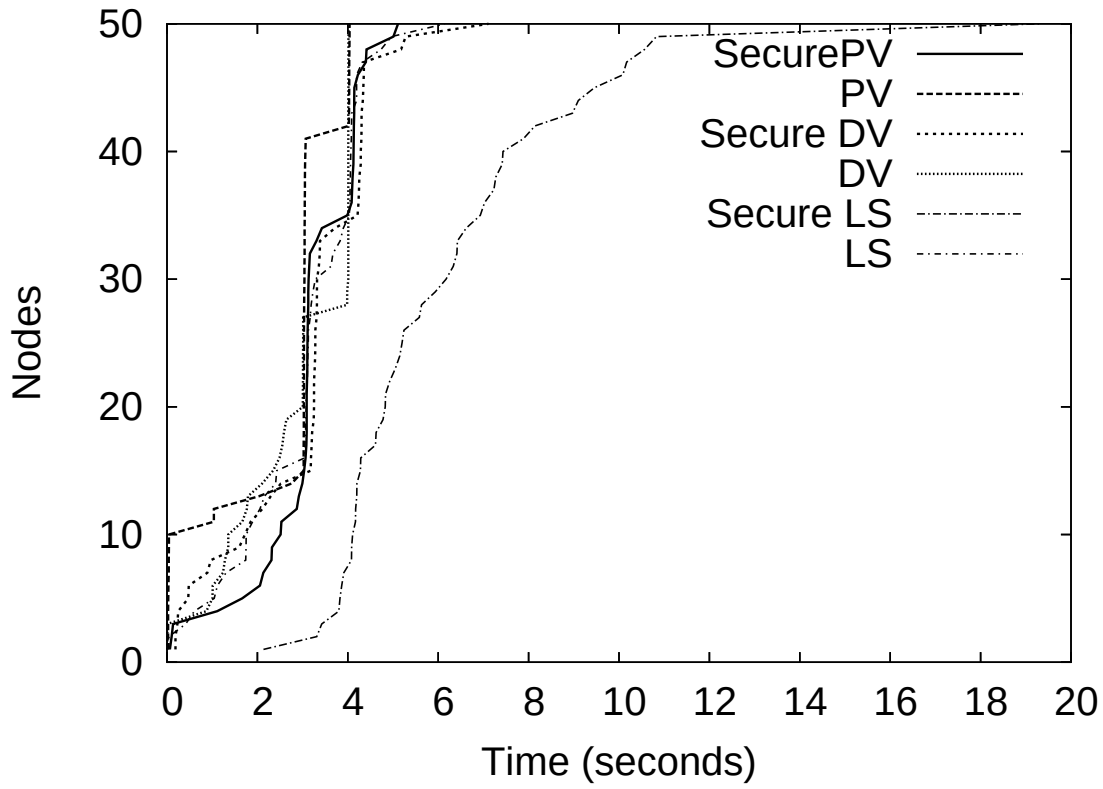


Figure 6.8: Comparison of time to convergence of secure and insecure versions of common routing protocols - Distance vector, Path vector, and Link state.

the nodes come up simultaneously. Initially nodes perform reliable discovery. Routing is begun 30 seconds after boot-up. A link between any pair of nodes has unit cost. We measure the time taken for a node to obtain a finite cost path to all the other nodes in the network. Figure 6.8 shows that the secure versions of the routing protocols perform nearly as well as the insecure versions. The median and 90th percentile convergence times are almost equal except for secure link-state routing which requires a larger number of messages before it discovers paths to nodes in the network. These results show that once reliable communication is established, the subsequent complexity of propagating routing updates in a secure manner is marginal.

6.6 Summary

In this chapter, we have addressed the question of securing routing protocols in a completely decentralized manner. Based on the direct relationship between secure routing and the reliable communication problem, we describe the design of a reliable communication toolkit that provides a generic set of primitives to enable decentralized security for a variety of routing protocols.

To briefly recap, Figure 6.9 presents a complete summary of our theoretical results on the reliable communication problem in unknown networks and the basic building blocks essential for achieving these results. The theoretical aspects of the reliable communication results were presented earlier in Chapters 4 and 5. The reliable communication toolkit implements the basic building blocks shown in the Figure 6.9. The reliable communication toolkit exports generic primitives which are protocol agnostic, that can be used to secure different routing protocols in a decentralized manner. Specifically, we show how these primitives can be integrated to secure three standard routing protocols:

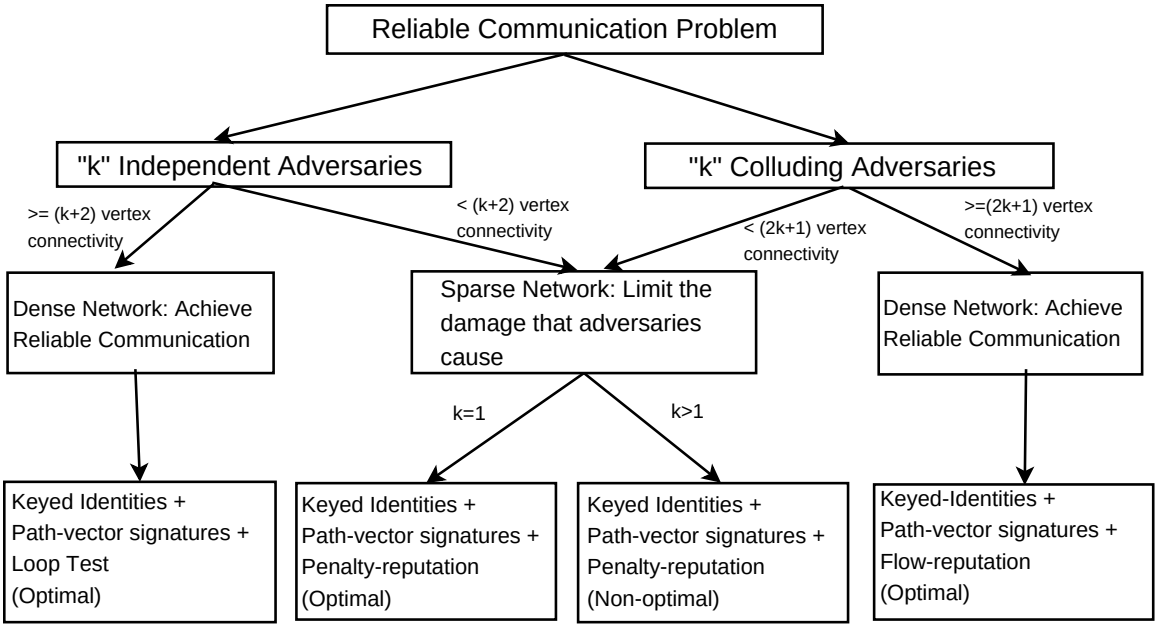


Figure 6.9: Summary of key results on reliable communication and the set of reliable communication primitives in the toolkit

link-state, distance-vector and path-vector routing. Finally, using a detailed performance evaluation of the toolkit, we show that cost of reliable communication is marginal. This demonstrates the fact that the reliable communication toolkit is feasible, practical and not expensive.

Chapter 7

Addressing the Data Integrity Threat to the Domain Name System

“ Marilyn Monroe wasn’t even her real name, Charles Manson isn’t his real name, and now, I’m taking that to be my real name. But what’s real? You can’t find the truth, you just pick the lie you like the best.”

– Marilyn Manson, ’Shock’ rock-star

The primary purpose of this chapter is to illustrate the broader applicability of the reliable communication concept beyond routing protocols. In this chapter, we describe the preliminary design of D-SecDNS, a decentralized security architecture that uses the reliable communication toolkit to address the data integrity security threat to the Domain Name System (DNS).

The DNS suffers from data integrity threats that arise due to configuration errors or malicious adversaries compromising name servers and propagating incorrect DNS responses to end-host queries. The data integrity problem is further exacerbated due to *transitive trust* relationships inbuilt in the

current DNS that introduces complex dependencies among name servers. These dependencies can enable a single compromised DNS server to affect the data integrity of not only its domain but also several other domains outside of its control.

Prior proposals [44, 48, 19] for addressing the data integrity problem of the DNS have relied on using a Public Key Infrastructure (PKI) with a central authority. Our proposal, D-SecDNS, on the other hand, does not rely on any form of prior key distribution mechanism or a central authority.

The rest of the chapter is organized as follows. In Section 7.1, we provide a brief overview of the Domain Name System. Next, we describe the data integrity security threat associated with the DNS in greater detail in Section 7.2. This is followed by a description of the different related work that have addressed the DNS data integrity threat in Section 7.3. We also briefly explain some of the pitfalls of prior work. Later, in Sections 7.4, we elaborate on the design principles and the architectural design of the D-SecDNS architecture. In Section 7.5, we describe the security guarantees that D-SecDNS can provide, show preliminary results in Section 7.6 to illustrate the feasibility of this architecture and present our conclusions from this study in Section 7.7.

7.1 Domain Name System: Overview

The Domain Name System (DNS) is an Internet directory service that resolves host names into IP addresses. The name-space of different host names is hierarchically partitioned into *domains* where each domain has several subdomains within the hierarchy. For example, *cs.berkeley.edu* is a subdomain of *berkeley.edu* which in turn is a subdomain of the top-level domain *edu*, which falls under the global root domain. Each domain is associated with a set of *authoritative nameservers*

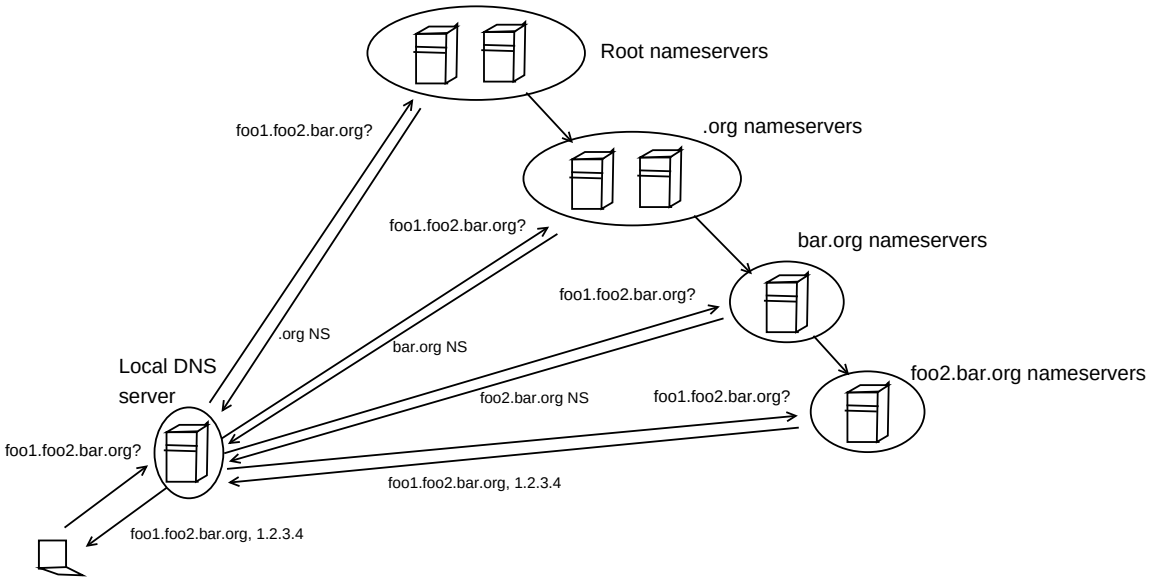


Figure 7.1: An example illustrating the DNS lookup process where the local DNS server iteratively contacts nameservers in the DNS hierarchy to resolve a DNS name.

that serve names within that domain. At the top of the DNS hierarchy are *root nameservers* and the authoritative nameservers for *top-level domains* (TLDs). The top-level domain name-space consists of generic TLDs (gTLD), such as *.com*, *.edu*, and *.net*, and country-code TLDs (ccTLD), such as *.uk*, *.tr*, and *.in*.

DNS uses a delegation based architecture for name resolution [86, 87]. Clients resolve names by following a chain of authoritative nameservers, starting from the root, followed by the TLD nameservers, down to the nameservers of the queried name. For example, the name *www.cs.cornell.edu* is resolved by following the authoritative name-severs of the parent domains *edu*, *cornell.edu*, and *cs.cornell.edu*. Following a chain of delegations requires additional name resolutions to be performed in order to obtain the addresses of intermediate nameservers. Each additional name resolution, in turn, depends on a chain of delegations.

DNS lookup process: The DNS lookup process is illustrated in detail in Figure 7.1. An end-host that wants to resolve the IP address corresponding to the host name *foo1.foo2.bar.org* performs the following steps:

1. Each end-host is initially configured with a set of local DNS servers. The end-host contacts one of the local DNS servers to resolve the name.

2. If the local DNS server has a cache entry corresponding to the name or is the authoritative nameserver for the domain *foo2.bar.org*, it provides a direct response to the end-host.

3. If not, the local DNS server contacts one of the root-servers which redirects the local DNS server to an authoritative nameserver corresponding to *bar.org*.

4. The local DNS server is repetitively redirected within the DNS hierarchy until it obtains the final response from an authoritative server of *foo2.bar.org* domain.

5. The local DNS server caches this response and also forwards it to the end-host.

7.2 The DNS data integrity threat

In *"Lives and Doctrines of Eminent Philosophers"*, one of the few ancient manuscripts to have survived since the 3rd century A.D., Diogenes Laertius poses the following famous question: *Could you tell me what was Plato's real name?* This question still remains unanswered for several centuries. Some say that Plato's real name was Aristocles, and that Plato was a nickname, meaning "the broad".

The same is the case with the DNS today. An end-host that queries for a DNS name has no proof

that the IP address provided as a response is indeed the address corresponding to the name. This is the *data integrity* problem associated with the current DNS.

The DNS forms a critical component of the Internet infrastructure which provides the essential service of name translation. Users accessing hosts on the Internet rely on the correct translation of host names to IP address by the DNS. However, the current design of DNS is susceptible to the *data-integrity* security threat whereby an attacker or a misconfigured nameserver can propagate incorrect DNS responses to name resolution queries. DNS data integrity can be affected due to configuration errors (*e.g.*, incorrect table entries) or various types of attacks launched by adversarial nodes. Once data integrity of name resolution is compromised, an attacker can redirect legitimate traffic to fake destinations. For example, the RSA Security web site was hijacked by spoofing DNS tables [1] wherein the attacker redirected all traffic bound to the RSA Security's original page to a fake destination by manipulating the corresponding DNS entry.

Now, we elaborate on three common ways through which the DNS data-integrity can be affected:

- *Configuration errors:* Several DNS entries in different tables are manually input creating the possible of data integrity problems due to incorrect entries. A recent study by Passas *et al.* [96] shows that 15% of DNS zones suffer from a specific misconfiguration called *lame delegation* in which the parent of a DNS zone points to wrong nameservers for the child zone. In the same study, they found that nearly 2% of zones have a *cyclic zone dependency* where information required to resolve a name in zone X depends on zone Y which in turn depends on zone X.

- *nameserver compromises:* Several DNS servers continue to execute unpatched, vulnerable

versions of DNS software making them easy targets for server compromises [26]. One software vulnerability of BIND is said to have affected a large number of DNS servers. An attacker that gains control over a nameserver can modify the DNS entries and thereby affect DNS data integrity.

- *Cache poisoning:* For performance reasons, every DNS server caches DNS responses for future use. DNS cache poisoning [48, 21, 111] occurs when a DNS server accepts and uses incorrect information from a host that has no authority giving that information. One mechanism for cache poisoning is for an attacker to initiate a DNS query to a DNS server X for *foo.attacker.net* where the attacker controls the nameserver Y for *attacker.net* domain. When X contacts Y to resolve *foo.attacker.net*, Y can respond with its entire set of DNS records in its cache (not just for the query) and poison X's cache with bogus DNS records (provided X caches Y's responses). Nearly $25-30\%$ of DNS servers are vulnerable to the cache poisoning attack [44].

7.2.1 Transitive trust relationships

The DNS data integrity problem is further exacerbated by the presence of transitive trust relationships in the current DNS model. DNS uses nameserver delegations to resolve host-names to IP address. This delegation mechanism induces complex dependencies between names and nameservers which can lead to a highly insecure system. A DNS lookup can be redirected to a nameserver *outside the control* of the primary domain. For example, while *cornell.edu* has 20 nameservers, only 9 of these belong to *cornell.edu*. Several nameservers that are outside the administrative domain of Cornell have indirect control over Cornell's name-space. In this case, *cornell.edu* depends on

rochester.edu, which depends on *wisc.edu*, which in turn depends on *umich.edu*. While Cornell directly trusts *cayuga.cs.rochester.edu* to serve its name-space, it has no control over the nameservers that *rochester.edu* trusts.

The transitive trust relationships lead to indirect dependencies between nameservers that can severely affect data integrity. The *delegation graph* associated with a domain consists of the transitive closure of all nameservers involved in the resolution of a given name associated to the domain. The nameservers in the delegation graph of a domain name form the *trusted computing base* (TCB) of that name. Compromise of any nameserver in the trusted computing base of a name can lead to a hijack of that name. Previous work by Ramasubramanian *et al.* [104] shows that the resolution of a domain name depends on a large trusted computing base of 46 servers on average, not including the root servers. Of this, only 2.2 servers on average are directly designated by the name-owner. Additionally, their work also shows that nearly 30% of domain names can be hijacked by compromising merely two servers.

The risk of domain hijack due to transitive trust relationships is further aggravated by the way DNS uses time-to-live (TTL) for each record. A compromised server providing incorrect records can associate long TTLs with the bindings ensuring that the incorrect records are cached and used for a longer period of time.

7.3 Prior work on DNS data integrity

DNSSEC [44, 19, 48, 111] is one of the first and foremost solutions that has been proposed to address the data-integrity security threat of DNS. DNSSEC provides a set of extensions to DNS

to perform three basic functions: (a) origin authentication of DNS data, (b) data integrity and (c) authenticated denial of existence.

DNSSEC uses a public-key infrastructure to achieve public-key distribution using which it can address DNS data integrity problems. DNSSEC relies on cryptography to ensure authenticity and integrity. DNS servers supporting DNSSEC use public-key cryptography to sign DNS responses. Every DNS query and response is associated with a transaction signature to authenticate communication between DNS servers as well as to authenticate the integrity of the response.

The DNSSEC approach has several shortcomings. First and foremost, DNSSEC relies on a public-key infrastructure for signing DNS responses. The use of a PKI induces a serious deployment hurdle since enforcing an Internet-wide PKI is fraught with the danger of non-adoption. Unlike the Internet routing case, the adoption can be easier in the DNS setting since the PKI hierarchy matches the DNS hierarchy. Apart from deployment hurdles, using a PKI induces additional problems. First, DNSSEC requires DNS servers to perform public-key operations for every DNS response. This can induce a heavy computational overhead at each server. Second, one drawback of the PKI hierarchy matching the DNS hierarchy is that if the authoritative name servers associated with a domain are compromised, then all subdomains associated with it are also affected. Hence, in DNSSEC, it is not possible to provide *islands of security* where one can secure a child zone though the parent zone may be insecure. Third, it is hard to perform *secure delegation* in DNSSEC. In other words, how does one delegate authority from a parent to a child zone in DNSSEC? The zone-cuts (boundaries) between different parent and child zones are ill-specified in DNSSEC. Finally, the DNSSEC approach for providing authenticated denial is complicated and expensive. The current approach is for the nx-domain responses where the parent DNS server responds that the child zone does not

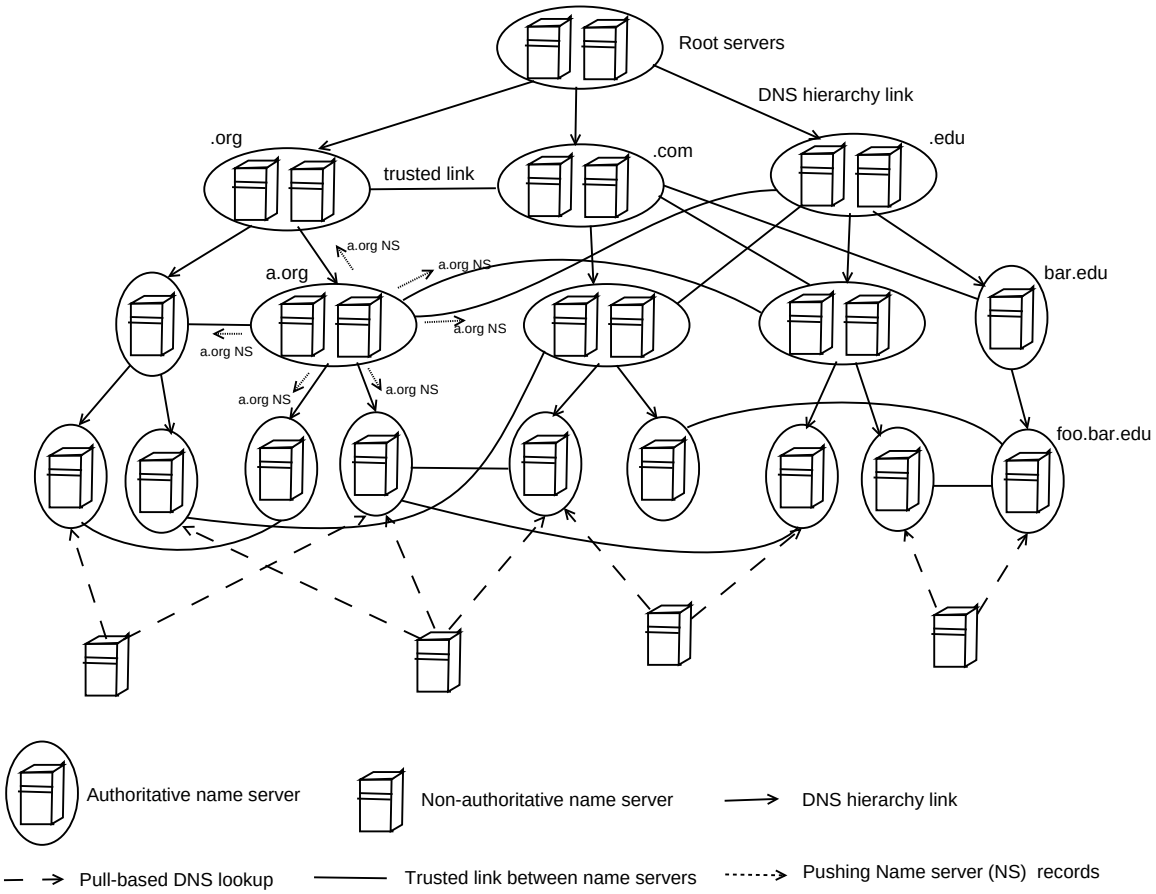


Figure 7.2: D-SecDNS architecture: Authoritative nameservers use additional trust links (apart from the DNS hierarchy) to set up a well-connected trust network between nameservers. Within this network, they disseminate nameserver records to other nameservers.

exist. Additionally, it is an complicated process for a parent zone to generate an authenticated denial response for every record.

7.4 D-SecDNS Architecture

In this section, we will describe the design of D-SecDNS, a decentralized architecture which uses the reliable communication concept to address the data integrity threat to the DNS.

The basic system architecture of D-SecDNS is illustrated in Figure 7.2. D-SecDNS makes an explicit distinction between authoritative and non-authoritative nameservers. An authoritative nameserver is a primary server that responds to DNS queries corresponding to a specific domain. A non-authoritative DNS server, on the other hand, acts primarily as a local DNS server within a domain which serves end-host requests and caches DNS responses from authoritative nameservers. Non-authoritative DNS servers are used primarily to improve the availability of the overall system.

Given the separation between authoritative and non-authoritative name servers, the key idea of D-SecDNS is to *use the reliable communication toolkit to propagate nameserver record information across authoritative nameservers.* We will now elaborate on this key design principle and how we achieve it.

7.4.1 Reliable communication between authoritative nameservers

The basic design principle used in D-SecDNS is to consider the network among the set of authoritative nameservers and establish a *reliable communication channel* between every pair of name servers. By doing so, every nameserver can directly contact the authoritative nameserver corresponding to a domain as opposed to using the traditional DNS lookup process. If a reliable communication channel can be established, this avoids many of the security threats that are induced by DNS redirection thereby reducing the scope of damage that a compromised nameserver can cause.

However, in order to establish a reliable communication channel between every pair of authoritative nameservers, we need to satisfy two basic constraints: (a) fixed-identity criterion; (b) connectivity constraint.

Meeting the fixed-identity criterion

We rely on the model of the current DNS architecture to satisfy the fixed-identity criterion. Each authoritative nameserver is associated with a set of trusted neighbors which include parent nameservers, child nameservers and delegation-based trusted nameservers. Whenever an domain adds a truster neighbor, it is the responsibility of the domain to ensure the correctness of the identity of the neighbor being bootstrapped into the system. In our threat model, we explicitly assume that *nameservers that are not compromised or misconfigured are not malicious i.e.,* a nameserver can turn malicious only when it is misconfigured or is compromised. Additionally, we assume that a configuration error does not affect the very identity of a node; while the nameserver records may be misconfigured, the identity of a node is not. Hence, once the fixed-identity is established at the time of forming a new trusted link, nameservers at either end are explicitly disallowed to modify their identities. This is to ensure that once the identity of a neighbor has been verified, it is enforced. Any modification to the identity is treated as breaking an existing trusted link. Therefore, once trusted links are established, the fixed identity criterion is enforced. If trusted links are established out of band, it is essential for the participating domains to initially verify the identities of their trusted neighbors.

Meeting the connectivity constraint

Previously in Chapter 4, we showed that a minimum vertex connectivity of $(2k + 1)$ is essential to achieve reliable communication in the face of k colluding adversaries. Hence, in order to establish reliable communication between authoritative name servers, it is essential to set up an underlying communication network between the nameservers which satisfies this vertex connectivity constraint.

Every link in this underlying network is required to be a *trusted* link between two nameservers. Hence, this network should be viewed as an extended version of the trusted computing base across domains where each authoritative nameserver has a trusted set of other authoritative nameservers as neighbors. The additional links established in the trusted computing base can be used as alternative channels for verifying the correctness of a DNS response. In the absence of a certificate authority, it is essential to have these alternate channel for verification.

However, achieving the $2k + 1$ vertex connectivity across the entire network of authoritative nameservers is a challenging problem. This would involve addressing two questions: (a) how does a nameserver identify other "trustworthy" nameservers to be direct peers with? (b) how does one achieve the connectivity constraint?

Establishing trusted links with neighbors: We propose two different strategies for building such a trusted computing base. The first is to use an *out-of-band* mechanism to establish relationships between different entities. Typically, each domain is under a specific administrative entity which enables the case for replicating the *public peering model* in BGP. In this model, a group different administrative entities together establish a public trusted relationship where every entity will have a trusted relationship with every other entity in the group. The problem with this model however is that smaller administrative entities have a bigger barrier to entry into such a system. The alternative model is to use an *in-band* mechanism where we use the existing DNS to establish new transitive trust relationships. Here, every pair of authoritative nameservers that establish a trusted link place a certain level of trust on the root to exchange path-vector signatures for the entire path from the root to each domain. However, in the in-band model, we expect each node to regularly modify its set of trusted links to reduce the potential damage that adversarial nodes can cause.

Connectivity constraint: Vertex-connectivity of a network is a global property and not a local one. To achieve a minimum connectivity of $2k + 1$ does not merely translate to each DNS server having $2k + 1$ trusted neighbors. For example, if every nameserver within the *.edu* domain only chooses other *.edu* nameservers, the vertex connectivity does not improve (except the paths through the DNS hierarchy). However, if the set of chosen trusted links is randomly distributed, then one can establish the connectivity using local constraint. The following result [27] establishes the requirement:

Theorem [Bollobas85]: *Consider a random graph G on n nodes where every edge is present with a probability p (*i.e., *average degree of a node is np). G is m-vertex connected with high probability if: $np > \log n + (m-1)\log\log n$.*

Using this result on random graphs, we can conclude that if the set of trusted neighbors of a nameserver are random distributed, each name server should choose roughly $\log n + 2k \log\log n$ nameservers as neighbors (n is the total number of nameservers) to achieve a $2k + 1$ vertex-connectivity for the trusted computing base with high probability. In reality, we intend to defend against a small set of adversaries (typical value of $k < 10$). For such small values, the number of additional trusted links required per nameserver is small as well as manageable.

7.4.2 Reliable dissemination of nameserver records

Previously, in Section 7.4.1, we described on how to construct a trusted computing base of authoritative nameservers that satisfies the fixed-identity and connectivity constraints. Once such a trusted computing base is setup, the next step is to use reliable communication toolkit to disseminate nameserver records amongst authoritative nameservers.

Corresponding to every domain, DNS maintains NS-records which contain the name to authoritative nameserver mapping and A-records which contain the authoritative nameserver to IP address mappings. Whenever an end-host performs a DNS query for an unknown name (not present in the local DNS server cache), the request traverses the DNS hierarchy to determine the NS-records and A-records of the authoritative name server corresponding to the destination domain. The current DNS uses a pull-based architecture wherein the NS-records and A-records are pulled for every query. In D-SecDNS, the goal is to disseminate the NS-records and A-records between nameservers.

Dissemination of nameserver records raises two important questions:

1. *Transitive ownership trust:* Can nameservers outside the control of a domain claim to be authoritative nameservers for that domain?
2. *Push vs pull:* The current DNS is built on the pull model where one fetches nameserver records. Does dissemination of nameserver records affect scalability?

We will now address these two questions in turn.

Explicitly disallow transitive ownership trust

The current DNS model of providing authority beyond the primary domain is fundamentally flawed from a security perspective. Today, the fact that *rochester.edu* nameserver is an authoritative nameserver for *cornell.edu* introduces an additional level of dependency between nameserver which can be used as a mechanism to attack the system. In such an inter-dependent system, a single compromised nameserver can not only affect its own domain but also inflict damage on any other domain which has a transitive trust relationship with it.

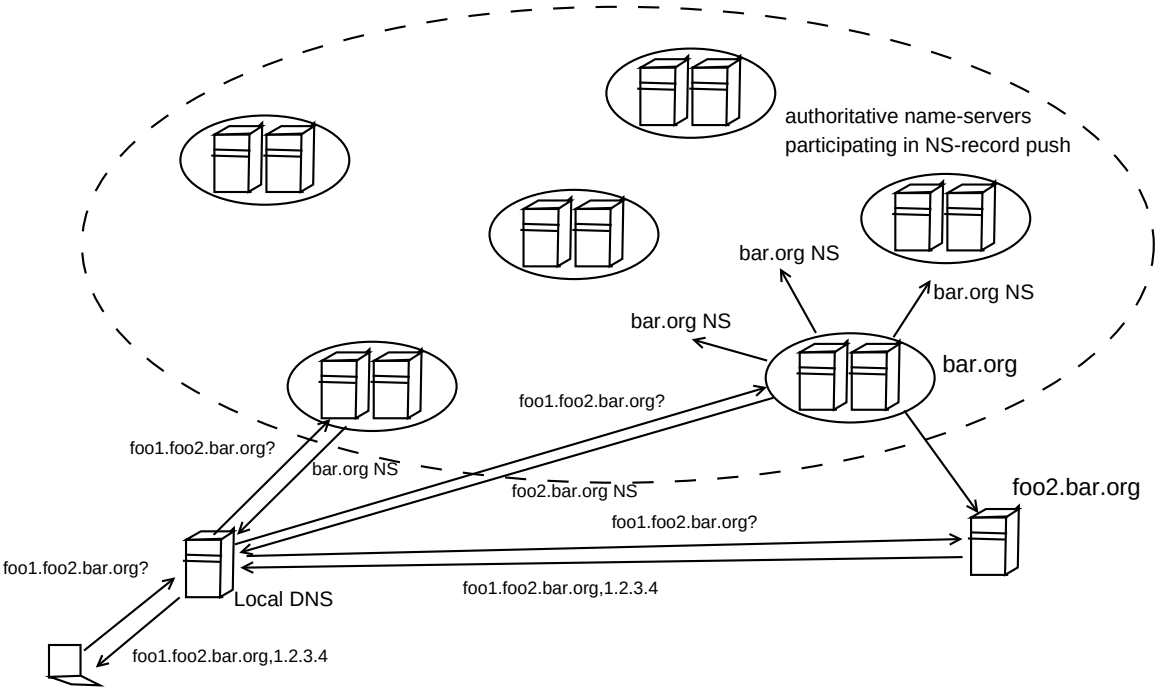


Figure 7.3: Hybrid push-pull model and the D-SecDNS lookup process example. Not every authoritative nameserver needs to participate in the process of proactive dissemination of NS records. In this example, *foo2.bar.org*'s authoritative nameserver does not participate in the NS-record push.

In D-SecDNS, we retain the authoritative nameservers corresponding to a domain to be only within the domain and not outside of it. While this may potentially reduce the number of authoritative nameservers corresponding to a domain, this is essential to restrict the damage that a single malicious server can cause. Hence, if an authoritative nameserver is compromised, it cannot directly affect data integrity on any other domain apart from its own domain. This also restricts the complex dependencies between nameservers introduced by transitive trust.

Hybrid push-pull dissemination model

The basic design principle is to use a *push-based* model where authoritative nameservers use the reliable communication toolkit to reliably broadcast their NS-records and A-records to other authoritative nameservers. From a security perspective, a push-based model has two significant advantages over a pull-based model. First, by reliably communicating the NS-records and A-records to other name servers, one can avoid the entire DNS redirection possibility and directly contact the authoritative nameserver corresponding to a domain. This prevents a variety of DNS redirection attacks that can be performed in a pull-based model. Second, a push-based model enables *off-line verification* of nameserver authority. In a pull-based model, the process of verifying the correctness of the authoritative nameserver corresponding to a domain for every query is a very expensive operation. To perform such a verification, the end-host needs to send the same query multiple times along different node-disjoint paths and test correctness across these. In a push-based model, this verification is an infrequent operation and can be performed offline. Off-line verification reduces the public-key cryptography burden that DNSSEC imposes for every DNS lookup.

However, one of the drawbacks of a push-based model is the amount of state that needs to be maintained by each authoritative server. In a pull-based model, the amount of state maintained by each nameserver is relatively low and each DNS record is fetched only on demand.

D-SecDNS can support a hybrid model between these two approaches where some authoritative nameservers may not participate in the push-based model and can revert to a pull-based model. An example of this is illustrated in Figure 7.3 where an authoritative nameserver of *foo2.bar.org* does not participate in the push model while its parent authoritative nameserver for *bar.org* participates

in the push-model. The nameserver for *foo2.bar.org* acts in the same fashion as a non-authoritative nameserver in passive mode pulling records from other authoritative nameservers (preferably from the parent). When an end-host issues a DNS query for *foo1.foo2.bar.org*, the push-based system directs the query to the nameserver of *bar.org* which then redirects (in the normal DNS fashion) to the nameserver of *foo2.bar.org*.

The hybrid push-pull model has better flexibility and scalability at the cost of slightly inferior security properties to the pure push-based model. The hybrid model has flexibility in determining the set of authoritative nameservers which participate in the push-based system where nameserver records are distributed. Hence, in a hybrid model, the size of the push-based system determines the amount of state maintained at each nameserver and the amount of nameserver information being disseminated. This in turn directly affects the scalability of the system

From a security perspective, a complete push-based model is preferable to a hybrid or a pull-based model. The reasons are two-fold. First, an authoritative nameserver that is not part of the push-based system is completely reliant on its parent domain. If the parent domain is compromised, the child zones also get affected. The same problem is present in DNSSEC. Finally, the overhead of public-key cryptography increases – for every DNS request, a DNS server needs to perform a verification process to test for validity. In a push-based system, this overhead is reduced by off-line verification of nameserver records which is a one-time operation.

7.4.3 Specific Design Issues

In D-SecDNS, the authoritative nameservers form a well-connected trusted computing base. Each authoritative nameserver claims ownership of a specific domain and propagates this ownership to

its neighbors which in turn propagate this information to their neighbors. Every ownership claim is propagated as a path-vector message along with a path-vector signature containing a chain of signatures from the authoritative nameservers along the path.

Three specific issues need to be addressed: (a) given two conflicting NS-records for the same domain name, how do we determine the validity of NS-records? (b) how do we determine whether a domain name exists? (c) How do we add new authoritative nameservers? We now describe how we deal with these questions:

Name-conflict resolution: Conflict resolution occurs in the same fashion as in the reliable communication toolkit. We use the *joint reputation metric* described in Chapter 6 to measure the trustworthiness of each domain name claim. To recall, the joint reputation metric uses a combination of the number of vertex-disjoint paths and penalties to determine the reputation of a domain claim. If the reputation of a name is more than k (a bound on the number of adversaries), the domain name claim is declared genuine. If however, the number of adversaries is more than k, the security property of determining domain names to be genuine does not hold any more.

Validity of a domain name: In addition to establishing a reputation for every domain name, every authoritative nameserver that obtains a nameserver claim associates each domain name with a *primary root path* through the DNS hierarchy. Consider an authoritative nameserver X and a domain name Y. The primary root path from Y to X refers to the path in the DNS hierarchy from the destination domain to a specific target domain. A specific example of one such primary root path is illustrated in Figure 7.4. The path-vector signature associated with the primary root path can traverse any of the authoritative name servers associated with each of the domains along the path. In our design, it is essential for each domain name to be associated with a primary root path for it to be

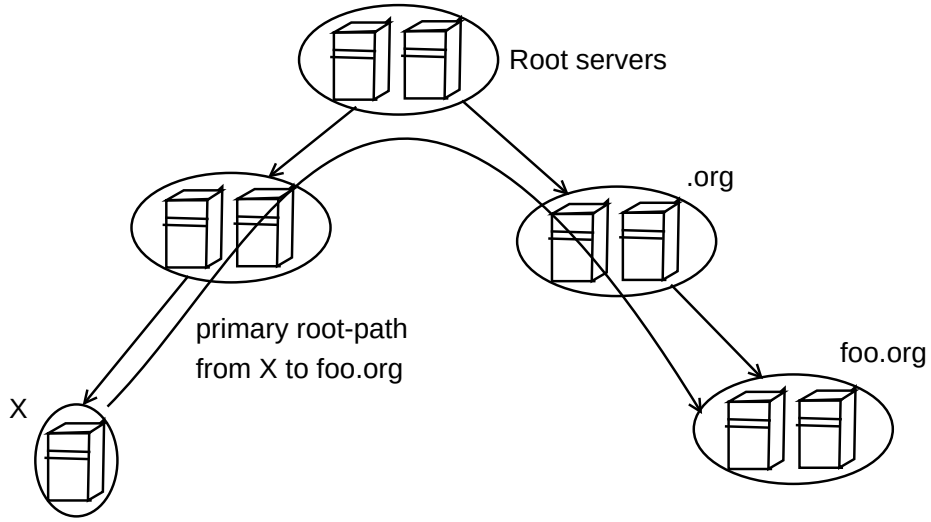


Figure 7.4: An example illustrating the primary root path.

considered a valid domain name. If an authoritative nameserver obtains a domain name claim that does not have a primary root path, such a domain is deemed to be non-existent. This reinforces the role of DNS hierarchy to determine the validity of a domain name. However, the primary root path is primarily used for determining the existence of a domain name but not to establish the validity of the NS-record and A-record corresponding to the domain. The validity of nameserver records is established based on the reputation computed by the reliable communication toolkit. This way, even if one of the authoritative nameservers of a parent domain is compromised, a child domain can establish the validity of its NS-records using alternate paths provided a primary root path claim corresponding to that domain name exists. Hence, D-SecDNS can provide security to child domains even if specific authoritative nameservers of the parent domain are compromised.

Adding a new authoritative nameserver: Adding a new authoritative nameserver is a relatively straightforward process. Consider a new nameserver X corresponding to the domain *xyz.abc.org*.

A new nameserver can only be introduced into the system by its parent. Hence, an authoritative nameserver corresponding to *abc.org* initiates a path-vector signature that propagates ownership information claiming X to be an authoritative nameserver for *xyz.abc.org*. This information needs to be pushed through the DNS hierarchy to all the other authoritative nameservers in the system which need to establish a primary root-path to the subdomain name *xyz.abc.org*. After this step, X can set-up trusted link (or initiate pre-existing trusted links) and propagate the ownership information through the additional trusted links. Every authoritative nameservers which can establish $(k + 1)$ disjoint paths to X in the trusted computing base will be able to verify that X is the genuine authoritative nameserver corresponding to *xyz.abc.org*.

7.4.4 The role of non-authoritative nameservers

In the D-SecDNS architecture, a non-authoritative nameserver acts in roughly the same fashion as a normal DNS server today with a few minor changes. Each such server maintains a cache of DNS responses. Upon each DNS request from an end-host, the server checks whether it has a cached response to the request. If not, the server needs to determine an authoritative nameserver corresponding to the target domain of the lookup and issue a DNS query to such a server.

A non-authoritative nameserver can determine an authoritative name server in two different ways: *passive* or *active* mode. In the passive mode, the non-authoritative nameserver can issue a query to any authoritative nameserver to determine the authoritative name server for a domain. In the traditional model, such a server would issue a query to the root server and issue repetitive queries to different authoritative nameservers within the hierarchy. However in D-SecDNS, given that NS-records and A-records are pushed to authoritative nameservers, any nameserver can act as a root

and provide the NS-record corresponding to a domain.

In the active mode, a non-authoritative nameserver can act as a *slave* to an authoritative nameserver and download the entire list of NS-records and A-records from the authoritative nameserver. However, by operating in slave mode, the non-authoritative nameserver only has the ability to download the NS-records from an authoritative server but cannot generate its own NS-records. The active mode enables a non-authoritative server to directly generate a query to a target domain nameserver without having to perform a lookup operation.

Validity of an NS-record: In both the passive and the active mode, it is essential to verify the validity of an NS-record. If a non-authoritative nameserver contacts exactly one authoritative nameserver, then it may obtain incorrect NS-record information if that server is compromised. To establish validity of an NS-record, it is essential to query multiple authoritative nameservers and compute a majority. To have provable guarantees in the face of k adversaries, it is essential to query $2k + 1$ different authoritative nameservers and compute a majority response. In the active mode, this verification is performed off-line and this removes the nameserver lookup part in name-resolution. Hence, this will improve DNS lookup performance at the expense of more nameserver state at a local DNS server. However, in the passive mode, verifying the validity of an NS-record is an expensive operation since this would require multiple NS-record lookup queries for every name resolution. In the passive mode, we initially perform a single NS-record lookup to one authoritative nameserver, but perform a lazy verification of the NS-record by querying other nameservers. Here, the server can aggregate several such queries into a single request in order to reduce the verification overhead. Hence, for the first DNS query to a target domain, we cannot guarantee validity of the NS-record if the authoritative nameserver contacted is compromised.

7.4.5 D-SecDNS name lookup process

Figure 7.3 illustrates the name lookup process in D-SecDNS. An end-host that wants to resolve the IP address corresponding to *foo1.foo2.bar.org* performs the following steps. First, each end-host is configured with a set of local DNS servers. The end-host contacts one of the local DNS servers with a DNS request. The local DNS servers examines its local cache and if the name is not present. If not, we have two different cases: (a) local DNS server is an authoritative nameserver; (b) local server is a non-authoritative server.

If the local DNS server is an authoritative nameserver, then it sends a request to the authoritative nameserver that has the longest suffix match to *foo2.bar.org*. Typically, the longest domain suffix should match *foo2.bar.org* in which case it should directly obtain a response. Otherwise, if the longest suffix is *bar.org* (which happens when the NS-record corresponding to subdomain *foo2.bar.org* is not pushed), the DNS server traverses the DNS hierarchy from *bar.org* to resolve the name.

If the local DNS server is an non-authoritative server in active mode, it performs the same process of querying the authoritative nameserver with the longest domain suffix match to *foo2.bar.org*. If in passive mode, this is preceded by a query to any authoritative name server to determine the longest suffix match. Upon receiving a response to a DNS query, the response is cached and forwarded to the end-host.

It is also important to note that from the end-host's perspective, D-SecDNS presents the same lookup interface as the traditional DNS. Hence, D-SecDNS does not require any modifications to end-hosts.

7.5 Security guarantees

D-SecDNS provides a practical security approach for addressing the data integrity threat to the DNS and we believe that D-SecDNS does not introduce new vulnerabilities over the underlying legacy DNS system. Overall, D-SecDNS provides the following security guarantees. First, it ensures that data integrity is preserved as long as the original authoritative nameservers are not compromised. Second, it alleviates the impact of compromises of legacy servers. Third, it eliminates the risk of domain hijacks introduced by subtle dependencies due to transitive trust. Finally, by disseminating DNS records it increases the resilience of the system by providing the ability to directly contact the authoritative nameserver of a domain as opposed to using redirection based lookup model. This also reduces the overhead of data-integrity verification by supporting off-line verification.

A fundamental issue in relying on DNS bindings obtained through any server in the system, not necessarily the servers authorized to disseminate the bindings, is a risk of accepting corrupt data. D-SecDNS uses the guarantees of the reliable communication toolkit, to ensure that the DNS bindings used are uncorrupted and match the DNS bindings originally advertised by the authoritative nameserver, as long as fewer than k malicious nodes in the system collude. A subtle security issue in D-SecDNS is however, in ensuring that the bindings are advertised by authoritative nameservers in the first place. Note that, it is possible for a number of malicious nameservers to pretend to be authoritative for a domain name, and there by introduce bogus bindings for that domain. Since the adversary is at a liberty to introduce any number of such bogus data sources, a simple majority voting is insufficient to authenticate the data, or more importantly will wrongly authenticate the bogus data. D-SecDNS thwarts this attack by verifying the authenticity of data sources with the

legacy DNS delegation path. Thus only DNS bindings introduced by a name server that is listed as authoritative for that domain by the parent zone are propagated by D-SecDNS.

The above scheme, however, does not provide protection when the attacker compromises the authoritative nameserver in the first place. D-SecDNS uses a simple technique to alleviate the effect of server compromises. Wherever there are multiple authorities for a domain name, it performs a simple majority voting to evaluate the authenticity of DNS data. Thus, unless the attacker takes over every authoritative nameserver for a domain, D-SecDNS can detect data integrity violations. However, majority voting cannot by itself separate corrupt data from correct data. Out-band intervention may be necessary to determine which nameservers have been attacked. Nevertheless, this scheme significantly raises the barrier to a successful attack and alleviates the impact of compromises of legacy DNS servers.

7.6 Feasibility study

In this section, we argue that the above described D-SecDNS architecture is practically feasible. From the data dissemination perspective, we quantify feasibility based on the following metrics: (a) effects of system size (number of DNS servers); (b) amount of information disseminated and update rate of information; (c) verification overhead. We will summarize the system size and update rate characteristics of the current DNS as quantified by various measurement studies and describe their implications on the D-SecDNS architecture.

System Size: Measuring the size of the existing DNS is a challenging problem in itself especially given that the current DNS uses a pull-based model. The pull-based model makes it difficult to

determine whether the entire space of nameservers and domains have been accurately covered by a measurement study. A recent large scale survey [103, 104] of DNS conducted at Cornell in July 2004 analyzes the properties of DNS bindings for about $600,000$ domain names obtained by crawling the Yahoo and Dmoz.org web directories. Their study indicates that about 530,000 distinct domains are served by about 160,000 different nameservers. A recent but more detailed survey [12] determined roughly 11 million domains of which several domains are being hosted by a single name server. As per their study, the DNS nameservers corresponding to *name-services.com* handled roughly $127,000$ domains each. Verisign's data indicates a linear growth in the number of registered domains in the past few years with roughly 10 million domains being registered each year (roughly $27,000$ per day). An alternate recent DNS study [8] discovered roughly $400,000$ nameservers. However, they also indicate that only a small fraction of these are authoritative nameservers. In summary, based on the existing measurement studies [103, 104, 8, 58, 12], the DNS supports roughly 100 million domains and has 7.5 million nameservers of which roughly $200,000 - 300,000$ servers are authoritative nameservers. The number of authoritative nameservers roughly matches with the number of IP prefixes within the BGP routing table.

In D-SecDNS, every authoritative nameserver maintains an entry corresponding to every authoritative nameserver but is not required to maintain the binding for every domain name. Hence, even though, the number of domains grows at the rate of tens of million domains every year, not each domain name to nameserver mapping is maintained by every authoritative nameserver. In fact, many subdomains of *.com* (or other top-level domains), often resolve to a specific end-host thereby making a *.com* nameserver as their authoritative nameserver. Thereby, a few nameservers end up being the authoritative nameserver for over $100,000$ domain names. Such domain name mappings

need not be pushed to all authoritative name servers and can be resolved by the longest suffix match which redirects them to the top-level domains. This hybrid approach drastically reduces the amount of state that needs to be distributed to (as well as stored at) each authoritative nameserver. In summary, we anticipate the number of domain name mappings that are widely distributed to be only a small fraction of the total number of domain names.

Update rate: The Cornell DNS study [104] also conducted an active study of the rate of change of DNS records performed by repeatedly polling the domain names every day and comparing the snapshots over a period of 1 week shows that the average amount of change per day is about 0.8%. Another study by Handley *et al.* [58], determined that roughly 0.5% of the domains change nameservers and about 0.1% of domains expire permanently everyday. Extrapolating to the entire DNS, approximately $420,000$ domains and $100,000$ domains change everyday. For the entire DNS, this translates to an update rate of 760 KB/hour or 1.6 Kbps, a relatively small bandwidth requirement. This assumes that all the domain name information is explicitly pushed; in reality, the required bandwidth should be lower than this quantity. Hence, the bandwidth requirement for propagating DNS updates is well within the bandwidth capabilities of existing nameservers. In face, this rate is orders of magnitude smaller than the total number of routing changes propagated in the Internet today; a single session reset can simultaneously trigger updates for $150 - 200K$ prefixes across a single hop.

Verification overhead: The reliable communication toolkit can verify the validity of roughly $40,000$ signatures per second. Hence, the verification overhead associated with proactive dissemination is negligible. Additionally, offline verification ensures that this verification process does not affect the performance of DNS lookup.

7.7 Summary

In this chapter, we described D-SecDNS, a decentralized architecture that addresses the data integrity security threat to the Domain Name System. D-SecDNS uses three basic ideas to address these security problems. First, D-SecDNS forms a trusted computing base across DNS servers that has a $2k + 1$ vertex connectivity and uses the reliable communication toolkit within this network to propagate DNS information reliably in the face of at most k adversarial nodes. Second, D-SecDNS does not delegate ownership of a domain outside it to prevent transitive trust problems. Finally, D-SecDNS distributes important nameserver information proactively using the reliable communication toolkit to provide the ability to pre-verify the authoritative name server for a domain.

D-SecDNS is one design point in the space of different mechanisms to secure the DNS. Four aspects of our design make it appealing. First, D-SecDNS is a completely decentralized architecture that does not rely in any form of central authority or a public key infrastructure. This enhances the deployability and acceptability of D-SecDNS. Second, D-SecDNS retains the existing notion of a DNS hierarchy yet can embed additional trusted links to this hierarchy to enable decentralized verification of DNS mappings. Third, apart from the need to push DNS mappings between nameservers, D-SecDNS retains many of the other functionalities of the existing DNS requiring only minor modifications for deployment. Finally, D-SecDNS does not require any modifications to end-hosts.

Chapter 8

Conclusions and Future Work

" Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning."

– Winston Churchill, November 1942

In this chapter, we conclude this dissertation with: (1) a summary of our contributions, (2) a description of the limitations of our solutions, and (3) a list of potential directions for future work.

8.1 Contributions

The main contribution of this dissertation is to address the question: *Under what constraints, can one can secure a routing protocol in a decentralized manner in the presence of adversaries?* This question was motivated by two related observations:

1. Today's Internet infrastructure is vulnerable to the threat of malicious routers hijacking Internet routes by propagating incorrect routing information. A single malicious router is capable

of hijacking a significant fraction of Internet routes by launching such an attack. Such incidents have happened in the past and have triggered Internet-wide outages. Hence, it is paramount to secure Internet routing from this threat.

2. Internet routing protocols are in dire need of decentralized security mechanisms given the deployment problems associated with a Public Key Infrastructure (PKI). Prior work [72, 112, 65, 92] on securing Internet routing protocols has primarily relied on the existence of a PKI with a central authority to enable verification of the correctness of routing messages.

One of the key observations that enabled us to address this problem is to map the secure routing problem to the *reliable communication problem in unknown networks* which represents a variant of the original reliable communication problem [45, 22]. In the original version of the reliable communication problem without signed messages (*i.e.*, no prior key distribution), an implicit assumption was that the entire network topology was known to every node in the network. However, in secure routing, this assumption does not hold; here, every node operates in an unknown network where every node is only aware of its neighbors and needs to discover a genuine route to every other node in the network.

The unknown network constraint makes the reliable communication problem a challenging one due to the absence of any form of node authentication. In an unknown network, adversarial nodes can generate incorrect routing information with non-existent nodes, edges and genuine nodes have no direct way of determining the truthfulness of a routing information. We address this challenge using the concept of *path-vector signatures* as described in Chapter 4 which enables a good node to classify truthful and incorrect information into different bins; however, they cannot always determine

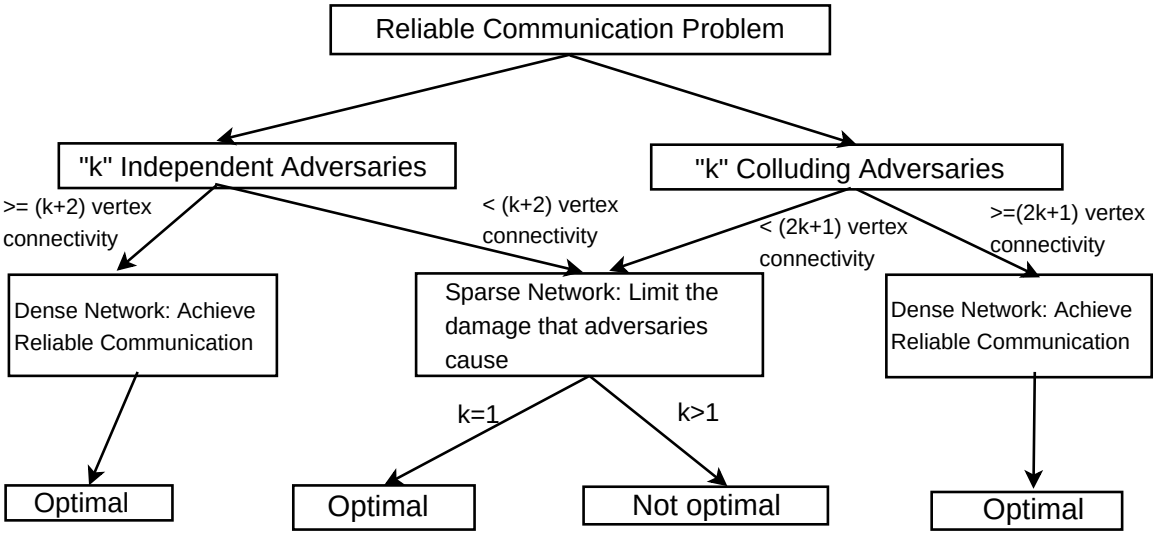


Figure 8.1: Summary of key results on reliable communication. In this figure, k represents the number of adversaries.

as to which bin contains the truthful information. Here, we show that if the network satisfies the *connectivity constraint* (*i.e.,* a minimum vertex connectivity of $2k+1$ in the presence of k colluding adversaries), then a good node can determine the truthfulness of routing information.

8.1.1 Summary of theoretical results on reliable communication

In this dissertation, we establish the following theoretical results (Chapter 4 and 5) on the reliable communication problem in unknown networks as summarized in Figure 8.1:

1. A network needs to satisfy the *fixed-identity criterion* where every node has a unique identity which it cannot fake. If this criterion is not met, then we show the existence of unknown networks where a single adversary is sufficient to disrupt reliable communication irrespective of the connectivity of the network.

2. Along with the fixed-identity criterion, we require a minimum level of network connectivity to achieve reliable communication. To handle k colluding adversaries, we require a minimum vertex connectivity of $2k + 1$ while for k independent adversaries, we only require a vertex connectivity of $k + 2$.

3. If the connectivity constraint is not met, we show that nodes can use *penalty-based filtering* as a defense strategy to limit the potential damage that adversaries may cause. We show that penalty-based filtering is the optimal defense strategy in the presence of a single adversary.

4. In Internet-like topologies which are modeled by power-law random graphs, we show that the cumulative damage that a single adversary can cause is very small ($O(n^{1/\alpha} \log^{\beta} n)$) where $2 < \alpha < 3$ is the power-law random graph parameter and β is a constant) compared to the size of the network (comprising n nodes).

8.1.2 Applying reliable communication theory in practice

In this dissertation, we demonstrate three different real-world applications on how the reliable communication concept theory can be applied in practice to achieve decentralized security:

1. **Listen and Whisper: Security Mechanisms for BGP** (Chapter 3): We propose Listen and Whisper as two mechanisms to secure the data plane and control plane of the Border Gateway Protocol (BGP). Whisper checks for consistency of routing advertisements in the control plane and Listen detects non-functional routes in the data plane by checking whether the data sent along routes reach the intended destinations. Whisper forms the basic building block for the path-vector signature construction used to solve the reliable communication

problem. Whisper along with penalty-based filtering is a direct application of the reliable communication theory in sparse networks (Chapter 5) where the goal is to limit the damage that malicious routers can cause. Listen presents a simple solution to the relaxed version of the data-plane secure routing problem described in Chapter 2. While one could apply the reliable communication theory to address the rigid version of the data plane secure routing problem, we chose not to do so since such a solution becomes expensive in practice.

2. **Reliable Communication Toolkit** (Chapter 6): The reliable communication toolkit is a system implementation of the basic building blocks essential to achieve reliable communication. This toolkit exports a generic set of primitives that can be integrated with various forms of routing protocols to secure them in a decentralized manner. In Chapter 6, we illustrate how this toolkit can be integrated with three basic routing protocols: link-state routing, distance-vector routing and path-vector routing. Using a detailed system evaluation of the toolkit, we show that reliable communication is feasible, practical and efficient from a systems perspective. Specifically, we show that the additional overhead of integrating the toolkit with routing protocols is minimal.

3. **Addressing Data Integrity Threat to the DNS:** (Chapter 7) The principle of reliable communication has broader applicability beyond routing protocols. To illustrate this, we present the design of D-SecDNS, a decentralized security architecture to address the data integrity security threat to the Domain name System (DNS). The basic idea in D-SecDNS is to use the reliable communication toolkit to establish reliable communication channels between different name servers. Using this channel, each name server disseminates its name-server records to other authoritative name servers in a verifiable manner. Thereby, D-SecDNS can ensure

the integrity of a DNS query response unless an authoritative name server of a domain is itself corrupted.

8.2 Limitations

In this section, we will describe some of the limitations of our solutions.

Fixed-identity criterion: Our theoretical results establish the need for the fixed-identity criterion to achieve reliable communication which in turn restricts the applicability of our solutions. There are many real-world networks like dynamic peer-to-peer networks and mobile ad hoc networks which do not satisfy the fixed-identity criterion. Our techniques are not directly applicable to such networks.

Connectivity constraint: In addition to the fixed-identity constraint, a network needs to have a minimum connectivity of $2k + 1$ in the face of k colluding adversaries for reliable communication. This imposes a fundamental bound on the number of adversaries that any decentralized security mechanism can handle. Unfortunately, the practical value of k that can be supported for many such real world networks is small typically in the range $[0..10]$. In fact, there are many networks which cannot even handle a single adversary completely. For example, the Internet topology is only $1-$connected, while portions of the topology are well-connected. In such a network, it is fundamentally impossible to achieve "perfect" decentralized security in the face of a single adversary. The best result we show for sparse networks is the optimality of penalty based filtering defense strategy in the presence of a single adversary. However, the case of the optimal defense strategy for the case of multiple adversaries is open. In the future, we hope to address this gap. Our solution for

sparse networks is based on minimalistic assumptions: (a) apart from the fixed-identity criterion, we operate in an unknown network and make no other specific assumption about the network; (b) every node locally estimates reputation of nodes for routes without relying on any form of reputation feedback from other nodes. One can potentially design better security mechanisms by exchanging reputation feedback between nodes and obtain tighter security guarantees for sparse networks. If we relax the unknown network assumption where one can has partial information of the topology (based on prior history in the absence of adversaries), then one can potentially get better security guarantees.

Data-plane correctness verification: Verification of route correctness in the data plane is fundamentally a hard problem in the face of adversarial nodes along the data path especially given that such nodes can merely drop all data packets while acting as genuine nodes for probe packets used for testing correctness. Any data-plane probing mechanism that tests validity of a route can potentially be proven to be correct if the adversarial nodes can explicitly identify such packets. The hardness of the data-plane verification problem motivated us to design *Listen* as a simple data plane verification mechanism for BGP. Listen is also susceptible to the case of adversarial nodes propagating bogus acknowledgments to TCP connections. In such a case, Listen will detect the route to be functional while in reality it is not. Detecting such non-functional routes with active adversaries along the data path is fundamentally hard.

8.3 Future Directions

There are several interesting directions for future work that one could pursue based on the work presented in this dissertation. Some of the future directions that we present are open theoretical problems relating to the reliable communication problem while others have more of a systems perspective to them.

8.3.1 Compact path-vector signatures

The path-vector signature construction presented in Chapter 4 which forms the basis of reliable communication has a message length of nM bits where n is the length of the path and every individual signature is represented using M bits (the value of M should be at least $O(\log N)$ where N is the size of the identity space). The open research question is: *Can one construct a compact path-vector signature construction which satisfies the same security properties of the original construction with a message length of $O(M)$ bits?* This question is relevant due to two reasons. First, in the context of BGP, reducing the signature length has the practical implication that the signature can be easily integrated into the BGP update message using a single field in the route updates. BGP has a maximum message size of 4096 bytes, which imposes certain limitations on the signature length per message; if longer, a signature needs to be split across two or more route updates. Second, reducing the message length also reduces the complexity of reliable communication since this reduces the time complexity of propagating messages across links as well as reduces the storage requirements at every node.

The problem of compact path-vector signatures has been addressed in the context of Secure-BGP [72].

All these works [73, 132] leverage the basic idea of *aggregate signatures*, a seminal cryptographic construct developed by Boneh *et al*. [28] where a set of n distinct signatures on n distinct messages from n different users can be aggregated into a single short signature. However, this construction relies on the fact that all the public keys of the users are derived from a single master authority (equivalent to central authority of a PKI). This assumption does not hold in unknown networks since each node generates its own keyed identity.

8.3.2 Sparse networks with multiple adversaries

As described earlier in Section 8.2, the problem of determining the optimal defense strategy in the case for sparse networks in the presence of multiple adversaries (colluding or independent) is an open research problem. The case of independent adversaries should be much simpler to address than the case of colluding adversaries. The challenging aspect of this problem of multiple adversaries (colluding or independent) is to determine the *minimum damage* that a set of adversaries can cause without being detected by good nodes in the network *i.e.,* no decentralized security mechanism should enable a good node to be able to pinpoint any adversarial node with certainty over other good nodes in the network. The property we require is that a set of adversaries can trigger spurious announcements as long as they can remain indistinguishable from other good nodes in the network.

8.3.3 Complexity analysis in unknown networks

We need good models for analyzing the complexity of distributed algorithms in unknown networks. The problem is that since the network is unknown, there is no global notion of time. In Chapter 4, we used a simplistic model which indirectly enforced a global notion of time by establishing band-

width constraints along each link. However, this model is too restrictive. In the face of queuing delays at each node, a single adversarial node propagating an infinite number of spurious messages (one message per unit time) can make several simple distributed algorithms to have exponential complexity. For example, consider Lemma 4 described in Chapter 4. In this case, the minimum time required to propagate a single message between two nodes (separated by a path of length l) in the face of adversaries continuously injecting spurious messages has exponential complexity in l. The problem with this analysis is that an adversarial node injects an exponential number of spurious messages (one message per unit time where each new spurious message has a new fake identity embedded in it) that in the presence of nodes with infinite buffers causes exponentially long delays. A more realistic model would involve only a bounded number of spurious messages with distinct fake identities from each adversarial node. In summary, one requires better complexity models for unknown networks to analyze the complexity of distributed algorithms.

8.3.4 Secure network coding

The reliable communication theory can be used to address some open security problems in the space of *network coding*. The concept of network coding was introduced in a seminal work by Ahlswede *et al.* [102] where they show that the net utilization of a network can be improved if intermediary nodes in the network are allowed to encode packets. The general network coding problem can be stated as follows:

Network Coding Problem: Consider a directed acyclic graph $G = (V, E)$ with a source s and a set of receivers $T \subset V$. Given that every edge in G has a unit capacity, what is the minimum transmission rate achievable from s to all the receivers in T?

Ahlswede et.al. [102] showed that there exists a network code which can send information at rate C_{min} where C_{min} is the minimum *edge-cut* from the source to each of the receivers.

One variant of the network coding problem that has not yet been completely addressed is the problem of *secure network coding*. The specific question, we wish to address: *What is the optimal capacity of network coding in the face of adversarial nodes in the network?* Prior work on estimating the capacity of network coding in the face of adversaries [60], only consider the case of wire-tap adversaries (adversaries on the links) but not the case where *nodes are adversaries*. Additionally, these works assume that the topology is known.

The secure network coding that we intend to address can be stated as follows:

Secure network coding problem: Consider a graph $G = (V, E)$ where each node is aware of only its neighbors and every link has bi-directional unit capacity. A source s in G intends to multicast a stream to a set of receivers $T \subset V$. Given a bound k on the set of colluding adversaries, what is the minimum transmission rate achievable from s to T?

Reliable communication enables us to achieve decentralized key distribution which in turn can aid in secure network coding. If reliable communication is achievable in a network, we conjecture that one can achieve *error-free* capacity where the adversarial nodes can at best act as node failures by dropping packets.

We define the *adversary-free subgraph* of a graph to be the subgraph generated by removing all the adversarial nodes from the original graph. We denote $advf(G)$ to represent the adversary-free subgraph of a graph G. We conjecture the following result:

Conjecture: *Consider an undirected graph G with at most k adversaries and unit-capacity edges.*

Let $\hat{G}$ be any directed acyclic orientation of an undirected graph G containing a source s and a set of receivers T. The capacity of secure network coding in $\hat{G}$ is equal to the multicast capacity of $advf(\hat{G})$ if and only if G is $2k+1$ vertex connected. Otherwise, the secure network coding capacity is zero.

8.3.5 Reputation systems

The area of reputation-based systems is growing rapidly as an important area in distributed systems [106]. Marti and Garcia-Molina [82] present a good survey of different reputation mechanisms for P2P mechanisms. In reputation systems, nodes associate a *reputation metric* with every other node in the system that determines the level of trust that one node places on another. In the presence of conflicting information, the reputation metric has been used as the primary metric for decision making in different systems [68, 51]. The reliable communication toolkit described in Chapter 6 is one such reputation system where every node computes a reputation for every route and uses this metric for route selection.

One of the important open problems in reputation systems is: *how does one compute reputation in the face of adversarial nodes whose primary goal is to game the reputation system?* An exact answer to this question may be dependent on the system under consideration, the underlying assumptions of the system and the adversarial model. In the reliable communication toolkit, the reputation metric we developed can provide provable guarantees provided the fixed-identity criterion holds and we are given a bound on the number of adversaries. Traditional reputation systems [106] operate under the assumption that the system only comprises of long-lived entities that capture and distribute feedback between them for computing reputations.

Two additional constraints make the problem even more challenging. First, in *unknown networks* like P2P networks [82] and Internet routing [119], we operate under the constraint that nodes are not aware of other nodes in the system. In such environments, associating nodes with reputation is fundamentally hard; a recent result by Cheng and Friedman [37] establishes the non-existence of Sybil-proof reputation mechanisms where nodes can modifies their identities. Second, a dishonest party can lie about the reputation metric that it propagates to other nodes. In this case, a node needs to compute a reputation metric based on potentially incorrect feedback from other nodes.

The larger vision for future work on reputation systems is the hope to develop a set of foolproof reputation mechanisms that can be used as building blocks across different kinds of systems. Such a system could have widespread applicability in P2P networks, distributed data storage and routing protocols among many other distributed systems.

8.3.6 Centralized vs decentralized security

Another of the important unaddressed research problems in this thesis is the gap between centralized and decentralized security. If the network satisfies the connectivity constraint, then the security guarantees offered by decentralized security matches that of centralized security using a Public Key Infrastructure (PKI). This is because reliable communication essentially provides decentralized key distribution if the connectivity constraint is met. In sparse networks, centralized security offers significantly better security guarantees than decentralized security; however, the exact gap between the two approaches is unknown for the case of multiple adversaries. For the case of a single adversary, the gap between the two approaches can be established based on the results described in Chapter 5.

Another open research problem is to quantify the effectiveness of a hybrid approach with *multiple*

certificate authorities, a potential security solution that is deployable for Internet routing. To address this, we recently proposed HLP [118], a next-generation routing protocol replacement for BGP, where we argued the model of deploying a separate PKI rooted at each tier-1 ISP and requiring the tier-1 ISPs to mutually re-conciliate routing conflicts. The security guarantees that such an approach offers have not yet been well understood.

8.3.7 Verifiable transactions in federated databases

A federated database [113, 17] is a distributed database consisting of a collection of sub-systems each maintained and administered by different autonomous entities. A federated database is similar to the case of Internet routing with different autonomous systems. Many database transactions in federated databases involve multiple autonomous entities who may potentially not trust each other.

We define a transaction to be *verifiable* in a federated environment if upon execution of the transaction, every autonomous entity participating in the transaction obtains a proof that the transaction executed the exact set of operations that it was supposed to perform as specified before execution. One crude example of a verifiable transaction of Internet routing is whenever a data packet is routed in the network, every autonomous system along the path obtains a proof that the data was correctly routed to the destination. We believe, reliable communication can aid as a building block for verifiable transactions in federated databases. One of the open research problems here is to clearly articulate the space of verifiable transactions in federated environments and the essential set of building blocks needed to support them.

8.3.8 Routing in sensor and fixed-wireless networks

Sensor networks and fixed-wireless networks (*e.g.*, wireless backhaul networks) are two other classes of networks which we did not consider in this dissertation which satisfy the fixed-identity criterion. These devices can be inbuilt with tamper-proof identities in the firmware [124] to satisfy the fixed-identity criterion. Hence, the reliable communication toolkit can be applied to these networks for achieving secure routing and secure data dissemination (in sensor networks). In a recent work by Wacker *et al.* [126], they provide a new mechanism for establishing pairwise keys in sensor networks. The underlying techniques used in this work closely match with the techniques used in our reliable communication toolkit to achieve decentralized key distribution.

The additional challenges that one faces in the context of sensor networks is *low battery power* and *low computing power*. Public-key cryptography operations are known to be very expensive for these tiny devices as illustrated by a recent work by Karlof and Wagner [69] on sensor network security. Hence, an open research question is to build practical and energy-efficient security mechanisms for achieving secure routing in sensor networks.

Bibliography

[1] Crackers cripple RSA Server. http://www.computeruser.com/newstoday/00/02/15/news2.html/.

[2] Habitat Monitoring on Great Duck Island. http://www.greatduckisland.net/.

[3] Internet Corporation for Assigned Names and Numbers. http://www.icann.org/.

[4] Internet Engineering Task Force. http://www.ietf.org/.

[5] Internet routing registry. http://www.irr.net/. Version current January 2003.

[6] Ivy - A Sensor Network Infrastructure for the College of Engineering. http://www-bsac.eecs.berkeley.edu/projects/ivy/.

[7] libpcap utility. http://sourceforge.net/projects/libpcap.

[8] The measurement factory dns survey. http://dns.measurement-factory.com/surveys.

[9] Microsoft port 1433 vulnerability. http:/lists.insecure.org/lists/vuln-dev/2002/Aug/0073.html.

[10] Oat Systems. http://www.oatsystems.com/.

[11] Ripe ncc. http://www.ripe.net.

[12] The security space dns survey. http://www.securityspace.com/s_survey/data/man.200511/dnsop.html.

[13] Sprint IPMON project. http://ipmon.sprint.com/.

[14] The Sun Global RFID Network Vision. http://www.sun.com/software/solutions/rfid/.

[15] Trends in dos attack technology. http://www.cert.org/archive/pdf/DoS_trends.pdf.

[16] World Summit on the Information Society. http://www.wsis-online.net/.

[17] ADIBA, M., AND DELOBEL, C. The Problem of Cooperation between different D.B.M.S. In *Architecture and Models in Data Base Management Systems* (1977).

[18] AIELLO, W., CHUNG, F. R. K., AND LU, L. A random graph model for massive graphs. In *ACM STOC* (2000), pp. 171–180.

[19] ARENDS, R., AUSTEIN, R., LARSON, M., MASSEY, D., AND ROSE, S. Protocol Modifications for the Domain Name System Security Extensions. Request for Comments 4035, March 2005.

[20] ARKKO, J., AND NIKANDER, P. How to authenticate unknown principals without trusted parties. In *Proc. Security Protocols Workshop 2002* (Cambridge, UK, April 2002).

[21] ATKINS, D., AND AUSTEIN, R. Threat Analysis of the Domain Name System. Request for Comments 3833, August 2004.

[22] BEIMEL, A., AND FRANKLIN, M. Efficient Reliable Communication over Partially Authenticated Networks. In *Theoretical Computer Science* (1999), vol. 220, pp. 185–210.

[23] BEIMEL, A., AND MALKA, L. Efficient Reliable Communication over Partially Authenticated Networks. In *PODC* (2003).

[24] BEN-OR, M. Another Advantage of Free Choice: Completely Asyncrhonous Agreement Protocols. In *PODC* (1983).

[25] BERKELEY MILLENNIUM CLUSTER. https://www.millennium.berkeley.edu/PSI.

[26] BIND Vulnerabilities. http://www.isc.org/sw/bind/bind-security.php, February 2004.

[27] BOLLOBAS, A., AND THOMASON, J. Random graphs of small order. In *Random Graphs, Annals of Discrete Mathematics* (1985), pp. 47–97.

[28] BONEH, D., GENTRY, C., SHACHAM, H., AND LYNN, B. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *Proc. of EUROCRYPT* (2003), pp. 416–432.

[29] BONO, V. J. 7007 explanation and apology. `http://www.merit.edu/mail.archives/nanog/1997-04/msg00444.html`.

[30] CACHIN, C., KURSAWE, K., AND SHOUP, V. Random Oracles in Constantinople: Practical Asynchronous Byzantine Agreement using Cryptography. In *PODC* (2000).

[31] CAI, M., HWANG, K., KWOK, Y.-K., SONG, S., AND CHEN, Y. Collaborative Internet Worm Containment. In *IEEE Security and Privacy* (May 2005), vol. 3(3), pp. 25–33.

[32] CASTRO, M., DRUSCHEL, P., GANESH, A., ROWSTRON, A., AND WALLACH, D. Secure Routing for Structured Peer-to-Peer Overlay Networks. In *Proc. of Symposium on Operating Systems Design and Implementation* (Boston, MA, December 2002).

[33] CHANG, D. F., GOVINDAN, R., AND HEIDEMANN, J. Temporal and topological characteristics of BGP path changes. In *IEEE ICNP* (2003).

[34] CHANG, H., R.GOVINDAN, JAMIN, S., SHENKER, S., AND WILLINGER, W. Towards capturing representative AS-level Internet topologies. In *Journal of Computer Networks* (2004).

[35] CHANG, H., WU, S. F., AND JOU, Y. F. Real-time protocol analysis for detecting link-state routing protocol attacks. *ACM Transactions on Information and System Security (TISSEC) 4*, 1 (2001), 1–36.

[36] C.HENDRICK. Routing Information Protocol, 1988. RFC 1058.

[37] CHENG, A., AND FRIEDMAN, E. Sybilproof reputation mechanisms. In *Proc. of P2PEcon* (2005).

[38] CHEUNG, S. An Efficient Message Authentication Scheme for Link State Routing. In *Proc. of 13th Annual Computer Security Applications Conference (ACSAC)* (December 1997).

[39] CHUNG, F., AND LU, L. The Average Distance in Random Graphs with given Expected Degrees. In *Internet Mathematics* (2003), vol. 1(1), pp. 91–114.

[40] CISCO NETFLOW. http://www.cisco.com/warp/public/732/netflow/index.html.

[41] CLARKE, R. Conventional Public Key Infrastructure: An Artefact Ill-Fitted to the Needs

of the Information Society. `http://www.anu.edu.au/people/Roger.Clarke/II/PKIMisFit.html`.

[42] COX, R., MUTITACHAROEN, A., AND MORRIS, R. Serving DNS using a Peer-to-Peer Lookup Service. In *Proc. of International Workshop on Peer-to-Peer Systems* (Cambridge, MA, March 2002).

[43] DAVIS, D. Compliance defects in public key cryptography. In *Proc. 6th USENIX Security Symposium* (1996).

[44] DNSSEC. http://www.dnssec.net.

[45] DOLEV, D. The byzantine generals strike again. In *Journal of Algorithms* (1982), pp. 14–30.

[46] DOLEV, D., DWORK, C., WAARTS, O., AND YUNG, M. Perfectly secure message transmission. In *Journal of the ACM* (1993), pp. 17–47.

[47] DOUCEUR, J. The sybil attack. In *Proc. of IPTPS* (2002).

[48] EASTLAKE, D. Domain Name System Security Extensions. Request for Comments 2335, March 1999.

[49] ELLISON, C., AND SCHNEIER, B. Ten risks of PKI: What you're not being told about public key infrastructure. *Computer Security Journal 16*, 1 (2000), 1–7.

[50] FEAMSTER, N., BORKENHAGEN, J., AND REXFORD, J. Guidelines for interdomain traffic engineering. *ACM Computer Communication Review* (2003).

[51] FELDMAN, M., LAI, K., CHUANG, J., AND STOICA, I. Robust Incentive Techniques for

Peer-to-Peer Networks. In *Proc. of ACM Conference on Electronic Commerce (EC)* (June 2004).

[52] FELDMAN, P., AND MICALI, S. Optimal algorithms for byzantine agreement. In *ACM Symposium on the Theory of Computing* (1988), pp. 148–161.

[53] GAMAL, T. E. A public-key cryptosystem and a signature scheme based on discrete logarithms. In *IEEE Transactions in Information Theory* (1985), pp. 469–472.

[54] GAO, L., AND REXFORD, J. Stable internet routing without global coordination. In *IEEE/ACM Transactions on Networking* (2001).

[55] GIORDANO, S. Mobile ad hoc networks. In *Handbook of wireless networks and mobile computing* (2002), John Wiley & Sons, Inc., pp. 325–346.

[56] GKANTSIDIS, C., MIHAIL, M., AND SABERI, A. Congestion and Conductance in Power-law graphs. In *ACM SIGMETRICS* (2003).

[57] GOODELL, G., AIELLO, W., GRIFFIN, T., IOANNIDIS, J., MCDANIEL, P., AND RUBIN, A. Working around BGP: An incremental approach to improving security and accuracy of interdomain routing. In *Proc. of NDSS* (San Diego, CA, USA, February 2003).

[58] HANDLEY, M., AND GREENHALGH, A. The Case for Pushing DNS. In *Proc. of Fourth Workshop on Hot Topics in Networks (HotNets)* (November 2005).

[59] HEFFERMAN, A. Protection of BGP Sessions via the TCP MD5 Signature Option. Request for Comments 2385, August 1998.

[60] HO, T., LEONG, B., KOETTER, R., MDARD, M., EFFROS, M., AND KARGER, D. R. Byzantine Modification Detection in Multicast Networks Using Randomized Network Coding. In *Proc. of International Symposium on Information Theory (ISIT) 2004* (2004).

[61] HU, Y., JOHNSON, D. B., AND PERRIG, A. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proc. of ACM Mobicom* (2002).

[62] HU, Y., JOHNSON, D. B., AND PERRIG, A. SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks. In *Proc. of WMCSA* (June 2002).

[63] HU, Y., PERRIG, A., AND JOHNSON, D. B. Wormhole detection in wireless ad hoc networks. Tech. Rep. TR01-384, Department of Computer Science, Rice University, December 2001.

[64] HU, Y., PERRIG, A., AND JOHNSON, D. B. Efficient security mechanisms for routing protocols. In *Proc. of NDSS'03* (2003).

[65] HU, Y., PERRIG, A., AND M.SIRBU. Spv: Secure path vector routing for securing BGP. In *Proc. of ACM SIGCOMM* (2004).

[66] INTERNET ASSIGNED NUMBERS AUTHORITY. http://www.iana.org.

[67] ISHIGURO, K. Zebra routing software. http://www.zebra.org/.

[68] KAMVAR, S. D., SCHLOSSER, M. T., AND GARCIA-MOLINA, H. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proc. of World Wide Web (WWW)* (2003).

[69] KARLOF, C., SASTRY, N., AND WAGNER, D. TinySec: A Link Layer Security Architecture for Wireless Sensor Networks. In *Proc. of ACM SenSys* (November 2004).

[70] KENT, S., LYNN, C., MIKKELSON, J., AND SEO, K. Secure Border Gateway Protocol (S-BGP) – Real World Performance and Deployment Issues. In *Proc. of Network and Distributed System Security Symposium, (San Diego, California)* (2000).

[71] KENT, S., LYNN, C., AND SEO, K. Design and analysis of the Secure Border Gateway Protocol (S-BGP). In *Proc. of DISCEX '00* (2000).

[72] KENT, S., LYNN, C., AND SEO, K. Secure Border Gateway Protocol (Secure-BGP). *IEEE Journal on Selected Areas of Communications 18*, 4 (April 2000), 582–592.

[73] KILTZ, E., MITYAGIN, A., PANJWANI, S., AND RAGHAVAN, B. Append only Signatures. In *Proc. of International Colloquium on Automata, Languages and Programming (ICALP)* (July 2005).

[74] KUMAR, A., POSTEL, J., NEUMAN, C., DANZIG, P., AND MILLER, S. Common DNS Implementation Errors and Suggested Fixes. Request for Comments 1536, October 1993.

[75] LABOVITZ, C., MALAN, R., AND JAHANIAN, F. Origins of Internet routing instability. In *Proc. IEEE INFOCOM* (1999).

[76] LAMPORT, L., SHOSTAK, R., AND M.PEASE. The byzantine generals problem. In *ACM Trans. Program. Lang. Systems* (1982), pp. 382–401.

[77] LEVINE, M. BGP noise tonight? NANOG mail archives, October 2001. `http://www.merit.edu/mail.archives/nanog/2001-10/msg00221.html`.

[78] LYNCH, N. *Distributed Algorithms*. Morgan Kaufmann, San Francisco, 1996.

[79] MAHAJAN, R., WETEHRALL, D., AND ANDERSON, T. Understanding BGP Misconfigurations. In *Proceedings of ACM SIGCOMM 2002* (2002).

[80] MAHAJAN, R., WETHERALL, D., AND ANDERSON, T. Understanding BGP misconfigurations. In *Proc. ACM SIGCOMM Conference* (Pittsburg, August 2002).

[81] MAO, Z., REXFORD, J., WANG, J., AND KATZ, R. H. Towards an accurate AS-level traceroure tool. In *ACM SIGCOMM* (2003).

[82] MARTI, S., AND GARCIA-MOLINA, H. Taxonomy of Trust: Categorizing P2P Reputation Systems. In *COMNET Special Issue on Trust and Reputation in Peer-to-Peer Systems* (2005).

[83] MERKLE, R. Protocols for Public Key Cryptosystems. In *Proc. of IEEE Symposium on Security and Privacy* (1980).

[84] MISEL, A. S. Wow, AS7007! NANOG mail archives, April 1997. `http://www.merit.edu/mail.archives/nanog/1997-04/msg00340.html`.

[85] MIZRAK, A., CHENG, Y.-C., MARZULLO, K., AND SAVAGE, S. Fatih: Detecting and Isolating Malicious Routers. In *Proc. of IEEE Conference on Dependable Systems and Networks (DSN)* (June 2005).

[86] MOCKAPETRIS, P. Domain Names: Concepts and Facilities. Request for Comments 1034, November 1987.

[87] MOCKAPETRIS, P. Domain Names: Implementation and Specification. Request for Comments 1035, November 1987.

[88] MOCKAPETRIS, P., AND DUNLAP, K. Development of the domain name system. In *SIGCOMM* (1988), pp. 123–133.

[89] MOCKAPETRIS, P., AND DUNLOP, K. Development of the Domain Name System. In *Proc. of ACM SIGCOMM* (Stanford, CA, August 1988).

[90] MOY, J. *OSPF: Anatomy of an Internet Routing Protocol*. Addison-Wesley, 1998.

[91] MURPHY, S., AND BADGER, M. R. Digital Signature Protection of the OSPF Routing Protocol. In *Proc. of Networks and Distributed Systems Securiry Symposium* (April 1996).

[92] MURPHY, S., GUDMUNDSSON, O., MUNDY, R., AND WELLINGTON, B. Retrofitting security into Internet infrastructure protocols. In *Proc. of DISCEX '00* (1999), pp. 3–17.

[93] NG, J. Extensions to BGP to support Secure Origin BGP (sobgp). Internet Draft draft-ng-sobgp-bgp-extensions-00, October 2002.

[94] ORAM, A. Peer-to-peer: Harnessing the power of disruptive technologies, 2001.

[95] PADMANABHAN, V. N., AND SIMON, D. R. Secure traceroute to detect faulty or malicious routing. In *Proc. HotNets-I* (2002).

[96] PAPPAS, V., XU, Z., LU, S., MASSEY, D., TERZIS, A., AND ZHANG, L. Impact of Configuration Errors on DNS Robustness. In *Proc. of ACM SIGCOMM* (Portland, OR, August 2004).

[97] PAXSON, V., AND S.FLOYD. Wide area traffic: Failure of poisson modeling. In *Proc. ACM SIGCOMM* (1994).

[98] PEASE, M., SHOSTAK, R., AND LAMPORT, L. Reaching agreement in the presence of faults. In *Journal of the ACM* (1980), pp. 228–234.

[99] PEI, D., MASSEY, D., AND L.ZHANG. Detection of Invalid Routing Announcements in RIP protocols. In *Proc. of IEEE Globecom 2003* (December 2003).

[100] PERLMAN, R. *Network layer protocols with Byzantine robustness.* PhD thesis, Massachusetts Institute of Technology, August 1988.

[101] PRETTY GOOD PRIVACY (PGP). http://www.pgpi.org.

[102] R. AHLSWEDE AND N. CAI AND S.-Y. R. LI AND R. W. YEUNG. Network Information Flow. In *IEEE Trans. on Information Theory, vol. 46, pp. 1204-1216* (2000).

[103] RAMASUBRAMANIAN, V., AND SIRER, E. G. The Design and Implementation of a Next Generation Name Service for the Internet. In *Proc. of ACM SIGCOMM* (Portland, OR, August 2004).

[104] RAMASUBRAMANIAN, V., AND SIRER, E. G. Perils of Transitive Trsut in the Domain Name System. In *Proc. of ACM Internet Measurement Conference (IMC)* (Berkeley, CA, October 2005).

[105] RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., AND SHENKER, S. A Scalable Content-Addressable Network. In *Proc. of ACM SIGCOMM* (San Diego, CA, August 2001).

[106] RESNICK, P., ZECKHAUSER, R., FRIEDMAN, E., AND KUWABARA, K. Reputation Systems. In *Communications of the ACM* (December 2000), vol. 43(12).

[107] RIPE'S ROUTING INFORMATION SERVICE RAW DATA PAGE. http://data.ris.ripe.net/.

[108] RIVEST, R. L., SHAMIR, A., AND ADLEMAN, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. In *Communications of the ACM* (February 1978), vol. 21(2), pp. 120–126.

[109] ROWSTORN, A., AND DRUSCHEL, P. Pastry: Scalable, Decentralized Object Location and Routing for Large-scale Peer-to-Peer Systems. In *Proc. of IFIP/ACM International Conference on Distributed Systems Platforms* (Heidelberg, Germany, November 2001).

[110] SCHNEIER, B. *Applied Cryptography*. John Wiley & Sons, Inc., 1996.

[111] SCHUBA, C. Addressing Weaknesses in the Domain Name System Protocol, August 1993. Masters Thesis, Purdue University.

[112] SECURE ORIGIN BGP (SOBGP). ftp://ftp-eng.cisco.com/sobgp.

[113] SHETH, A. P., AND LARSON, J. A. Federated Database Systems for Managing Distributed, Heterogeneous and Autonomous Databases. In *Proc. of ACM Computing Surveys* (1990), pp. 183–236.

[114] SMITH, B., AND GARCIA-LUNA-ACEVES, J. Securing the Border Gateway Routing Protocol. In *Proc. Global Internet '96* (London, UK, November 1996).

[115] SMITH, B., MURPHY, S., AND ACEVES, J. G. L. Securing Distance Vector Routing Protocols. In *Proc. of Networks and Distributed Systems Security (NDSS) Symposium* (February 1997).

[116] STEWART, J. W. *BGP4: Inter-Domain Routing in the Internet*. Addison-Wesley, 1999.

[117] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, F., AND BALAKRISHNAN, H. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. of ACM SIGCOMM* (San Diego, CA, August 2001).

[118] SUBRAMANIAN, L., CAESAR, M., EE, C. T., HANDLEY, M., MAO, Z., SHENKER, S., AND STOICA, I. HLP: A Next Generation Inter-domain Routing Protocol. In *Proc. of ACM SIGCOMM* (August 2005).

[119] SUBRAMANIAN, L., KATZ, R. H., ROTH, V., SHENKER, S., AND STOICA, I. Reliable Broadcast in Unknown Fixed-Identity Networks. In *ACM PODC* (June 2005).

[120] SUBRAMANIAN, L., ROTH, V., STOICA, I., SHENKER, S., AND KATZ, R. H. Listen and Whisper: Security Mechanisms in BGP. In *Proceedings of ACM/ USENIX NSDI 2004* (2004).

[121] SUBRAMANIAN, L., S.AGARWAL, J.REXFORD, AND KATZ, R. H. Characterizing the Internet hierarchy from multiple vantage points. In *IEEE INFOCOM* (New York, 2002).

[122] THOMAS, R. `http://www.cmyru.com`.

[123] THOMPSON, K. Reflections on Trusting Trust. In *Communications of the ACM* (August 1984).

[124] TRUSTED COMPUTING GROUP. https://www.trustedcomputinggroup.org.

[125] UNIVERSITY OF OREGON ROUTEVIEWS PROJECT. http://www.routeviews.org/.

[126] WACKER, A., KNOLL, M., HEIBER, T., AND ROTHERMEL, K. A New Approach for

Establishing Pairwise Keys for Securing Wireless Sensor Networks. In *Proc. of ACM SenSys* (November 2005).

[127] WAN, T., KRANAKIS, E., AND OORSCHOT, P. V. S-RIP: A Secure Distance Vector Routing Protocol. In *Proc. of Applied Cryptography and Network Security (ANCS)* (June 2004).

[128] WANG, F. Y., AND WU, F. S. On the Vulnerability and Protection of OSPF Routing Protocol. In *Proc. of IEEE International COnference on Computer Communications and Networks* (October 1998).

[129] WHITE, R., AND FEAMSTER, N. Considerations in Validating the Path in Routing Protocols. Internet Draft, October 2003.

[130] YEGNESWARAN, V., AND BARFORD, P. Global Intrusion Detection in the DOMINO Overlay System. In *Proc. of Networks and Distributed Systems Symposium (NDSS)* (February 2004).

[131] ZHAO, B., HUANG, L., STRIBLING, J., RHEA, S., JOSEPH, A., AND KUBIATOWICZ, J. Tapestry: A Resilient Global-scale Overlay for Service Deployment. *IEEE Journal on Selected Areas in Communications 22*, 1 (January 2004), 41–53.

[132] ZHAO, M., SMITH, S. W., AND NICOL, D. M. Aggregated path authentication for efficient BGP security. In *Proc. of ACM Conference on Computer and Communications Security (CCS)* (November 2005).

[133] ZHAO, X., PEI, D., WANG, L., MASSEY, D., MANKIN, A., WU, S. F., AND ZHANG, L. An analysis of BGP multiple origin AS (MOAS) conflicts. In *ACM SIGCOMM IMW* (2001).

[134] ZHU, D., GRITTER, M., AND CHERITON, D. Feedback based routing. In *Proc. of HotNets-I* (October 2002).