TOWSON UNIVERSITY
OFFICE OF GRADUATE STUDIES

An Analysis of Susceptibility in Securing Multifaceted *Border Gateway Protocol* (*BGP*)

by

Kyoungha Kim

A Dissertation Presented to the faculty of Towson University
in partial fulfillment of the requirements for the degree of

Doctor of Science
In Department of Computer & Information Sciences

at the

TOWSON UNIVERSITY
Towson, Maryland 21252

(May 2016)

TOWSON UNIVERSITY

OFFICE OF GRADUATE STUDIES

DISSERTATION APPROVAL PAGE

This is to certify that the dissertation prepared by Kyoungha Kim entitled An Analysis of Susceptibility in Securing Multifaceted Border Gateway Protocol (BGP) has been approved by the dissertation committee as satisfactorily completing the dissertation requirements for the degree of Doctoral Science (D.Sc.) in Department of Computer & Information Sciences. The next page is the scanned original dissertation approval page.

Author .....

Kyoungha Kim

Department of Computer & Information Sciences

April 25, 2016

Certified by .....

Dr. Yanggon Kim

Dissertation Supervisor

Certified by .....

Dr. Robert Hammell

Dissertation Committee Member

Certified by .....

Dr. Alexander Wijesinha

Dissertation Committee Member

Certified by .....

Dr. Wei Yu

Dissertation Committee Member

TOWSON UNIVERSITY
OFFICE OF GRADUATE STUDIES

DISSERTATION APPROVAL PAGE

This is to certify that the thesis prepared by [INSERT Student's Name] Kyoungha Kim entitled [INSERT Title of Thesis] An Analysis of Susceptibility in Securing Multifaceted Border Gateway Protocol (BGP)

has been approved by the thesis committee as satisfactorily completing the dissertation requirements for the degree_ [INSERT Type of Degree] Doctor of Science in Information Technology
(for example, Doctor of Science)

Chairperson, Dissertation Committee Signature	YANGGON KIM	April 20, 2016
Committee Member Signature	ROBERT J. HAMMELL II	4/20/16
Committee Member Signature	ALEX WIJESINHA	4/20/16
Committee Member Signature	Wei Yu	4/20/2016
Committee Member Signature	Type Name	Date
Dean of Graduate Studies	Janet V DeLany	4-28-16

Type Name | Date

Acknowledgments

It is a great opportunity for me to write about "An Analysis of Susceptibility in Securing Multifaceted *Border Gateway Protocol* (*BGP*)." This field requires me to fully understand many *BGP* proposals, as well as the basic *BGP* protocol. My research is extremely challenging in that it could get just started after I looked into over hundreds of readings, especially for many standards. This paper includes most valuable assets that I found from my research experience more than three years. My proposals are based on those exhaustive background knowledge in such a way that it is necessitated for me to have knowledge of, at least, (i) network programming skills, (ii) entirely implementing a *BGP* speaker, (iii) the complete *Resource Public Key Infrastructure* (*RPKI*) and *BGPsec* standard, (iv) and facilitating related *BGP* implementations. Although I have tried hard and soul to propose relevant problems and solutions regarding the subject, any probable shortcoming, factual error, or mistaken opinion, might exist in this paper. However, I assure that this paper will be certainly immense importance for those who are interested in computer networks, not restricted to *BGP*. I hope that they can find comprehensiveness from this paper.

I acknowledge with deepest gratitude to my advisor, Dr. Yanggon Kim, who introduced me to computer networks. He is the biggest reason that I have chosen Towson university for my doctoral degree. He convincingly and infinitely conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. I would like to include a special note of warmly thanks to my family, especially to Mrs. Youngja Kang, for their unconditional love, support, and sacrifice for my dream. Without their wisdom and persistent help, I would not reach such a successful terminus for my dissertation paper and Ph.D. degree.

I also place on record, my sense of gratitude to one and all who directly

or indirectly have lent their helping hand in this study. Finally, thanks and appreciation are extended to future readers. I particularly have paid attention to the paper organization and presentation not to waste your precious time. Enjoy and thanks for your interest in this paper.

An Analysis of Susceptibility in Securing Multifaceted *Border Gateway Protocol* (*BGP*)

by

Kyoungha Kim

Submitted to Department of Computer & Information Sciences
on April 25, 2016, in partial fulfillment of the
requirements for the degree of
Doctor of Science
In Department of Computer & Information Sciences

Abstract

The *Border Gateway Protocol* (*BGP*), which is an inter-domain routing protocol, does not validate the origin and path information that are used to form *BGP* networks, resulting in many *BGP*-routing accidents. The *Resource Public Key Infrastructure* (*RPKI*) and *BGPsec* are remarked as solutions to provide the *Origin Validation* and *Path Validation* into *BGP*. The validation methodologies in those solutions are based on the *Public Key Infrastructure* (*PKI*) so as to offer the strong security level but produce significant computational overhead, especially in validating paths. In this paper, we contribute to developing *BGP* implementations for researchers to perform practical *BGP* simulations based on *RPKI* or *BGPsec*, as well as to analyze *BGP* data. Our implementations also introduce less computational overhead with the same capabilities.

Dissertation Supervisor: Dr. Yanggon Kim
Title: Professor

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 The Motivation to Research

This dissertation is designed to examine the security of a premature network protocol. In other words, any protocol that is constructed under not enough consideration of security can be investigated in the same manner. Given that the *Transmission Control Protocol/Internet Protocol* (*TCP/IP*) model is employed, our contribution is primarily associated with the *Internet (a.k.a., Network)* and *Host-to-Host Transport (a.k.a., Transport)* layer. In other words, most of our works provide network protocol analysis regarding low-level software, rather than hardware. Our analytical aspects of a network protocol can be categorized into four components, which are (i) *BGP Data Analysis* (ii) *Performance Analyzation* (iii) *Understanding of the current protocol* (iv) *Optimizing (or debugging) of the protocol*. In general, the *BGP Data Analysis* can be accomplished by capturing/decoding/verifying/investigating the contents of packets. The *Performance Analyzation* can be achieved by documenting timestamps of packets and monitoring hardware resources such as *Central Processing Unit* (*CPU*) and *Random Access Memory* (*RAM*). The specification and simulation of a protocol give rise to the third topic, the *Understanding of the current protocol* , from which the weakness analysis is additionally required in the *Optimizing (or debugging) of the protocol*.

Among various network protocols, we have chosen the ***Border Gateway Protocol* (*BGP*)** [1], which is an inter-*Autonomous System* (*AS*) routing protocol also known as an inter-domain routing protocol. Our decision depends on that the *BGP* is originally designed to operate in a trusted environment, and there are no internal mechanisms to protect the information it carries. The *Internet* is a global system of interconnected computer networks that utilize the standard *Internet Protocol* (*IP*) suite. Approximately 9.8 billion devices are linked to the *Internet* in 2015 and the number of connected devices would be increased to around 50 billion by 2020 [3]; the 9.8 billion is calculated from the estimated total number of things in 2015, which is 1.6 trillion, implying 0.6 percent penetration rate [4]. The connection between devices through the *Internet* usually facilitates, at least, one intra-domain routing protocol and inter-domain routing protocol. An inter-domain routing protocol exchanges routing and reachability information between domains. The domains are called ***Autonomous System*s (*AS*es)**, each of which is a set of routers under a single technical administration, using an *Interior Gateway Protocol* (*IGP*) and common metrics to determine how to route packets to their destinations (i.e., *AS*es). *AS*es are usually connected to multiple provider *AS*es to leverage fault tolerance [5]. Even when multiple *IGP*s and metrics are utilized, the administration of an *AS* appears to other *AS*es to have a single coherent interior routing plan and presents a consistent picture of the destinations that are reachable through it [1]. A peer in a different *AS* is referred to as an external peer while a peer in the same *AS* is referred to as an internal peer. ***Internal*** and ***External Gateway Protocol***, which is commonly abbreviated as *IGP* and *EGP*, are facilitated respectively for the communication between internal and external peers. Note that the peers are also referred to as *BGP Speaker*s. The *IGP* are *Interior Gateway Routing Protocol* (*IGRP*), *Intermediate System to Intermediate System* (*IS-IS*), *Open Shortest Path First* (*OSPF*), and *Routing Information Protocol* (*RIP*). The only *EGP* is *BGP*; note that the *BGP* in this paper is particularly *BGP*-4. The *BGP* can be

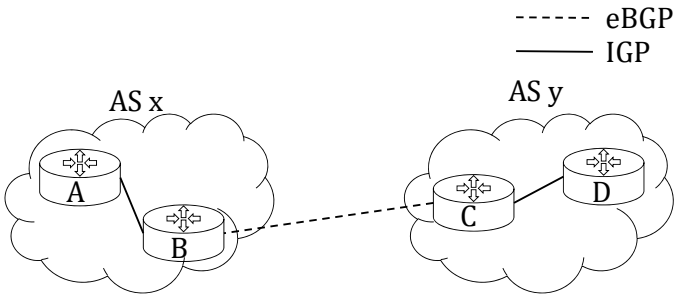


Figure 1-1: The Example of *eBGP*, *iBGP*, *IGP* in Use

categorized into ***External BGP*** **(*eBGP*)** and ***Internal BGP*** **(*iBGP*)**. While *eBGP*is leveraged to share the *EGP* information with the *BGP Speaker*s located in the other *AS*es, *iBGP*shares the information with the *BGP Speaker*s within the same *AS*. For example, Figure 1-1 depicts a simple *BGP* network with two *AS*es, each of which owns two *BGP Speaker*s. The *BGP Speaker B* learns routes to the speaker *C* and *D* from the border router *C* via the *eBGP* session between the speaker *B* and *C*. The same routes are educated to the *BGP Speaker A* from the speaker *B* via the *iBGP* session between them. The *iBGP* session is established by *IGP*. In summary, the *BGP* is utilized to form the *Internet* by connecting *AS*es based on "hearsay" and leaves much to be desired in security, resulting in high vulnerabilities. We contributed to inspect the various aspects of the protocol to secure *BGP* networks. We believe in that most methodologies we introduce can be applied to any network protocol that requires being protected.

1.2 The Overview of the *Border Gateway Protocol* (*BGP*)

1.2.1 The Allocation of *IP Prefix*es

The *Internet* consists of a large number of interconnected *AS*es [6], which exchange their routes using *BGP*. Therefore, the *BGP* plays a central role in making the *Internet* work by distributing and maintaining routing information.

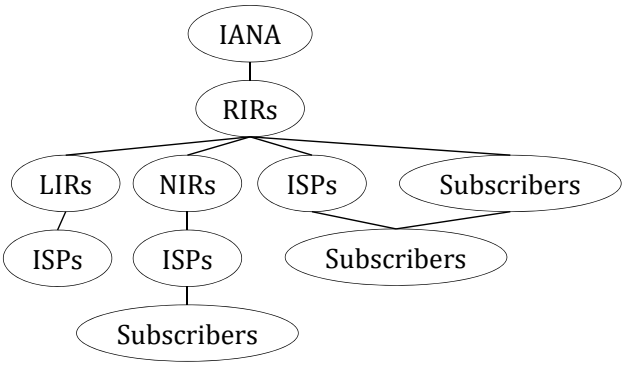


Figure 1-2: The Example of Allocating *IP Prefix*es

In particular, each *AS* can own one or more ***IP Prefix*es**, each of which is represented by the *Classless Inter-Domain Routing* (*CIDR*) [7] notation in *IP version 4* (*IPv4*) and *IP version 6* (*IPv6*); for example, 192.168.100.0/22 and 2001:db8:3c4d::/48. An *IP Prefix* is assigned to a source or destination device (i.e., an end device) for use on the *Internet*. The ***Internet Assigned Numbers Authority* (*IANA*)** [8] is responsible for the global coordination of the *Domain Name Server* (*DNS*), *IP* addressing systems, *AS* numbers (*ASN*s), and other *Internet* protocol resources. Both *IP version 4* (*IPv4*) and *IP version 6* (*IPv6*) addresses are generally assigned in a hierarchical fashion as illustrated in Figure 1-2. Speaking of which, the *IANA* allocates *IP* addresses from the pools of unallocated addresses to one of five *Regional Internet Registries* (*RIR*s) according to their demands. The *Internet* resources assigned to the *RIR*s can be redistributed to sub-organizations such as an *Local Internet Registry* (*LIR*) or *National Internet Registry* (*NIR*). The *Internet Service Provider* (*ISP*) that generally provides *IP* addresses to end users acquires those *IP* addresses from its parent organization (i.e., a provider) such as a *LIR* or *NIR*. The prefixes that have not been allocated to *RIR*s [8] are referred to as ***Bogon*s**, and the *IP* spaces that have been allocated to *RIR*s but reserved or maintained by *IANA* are called ***Full-Bogon*s**.

1.2.2 The *Autonomous Systems* (*AS*es)

The interconnections between *AS*es demonstrates a similar hierarchy to the *IP Prefix* allocation; namely, those two hierarchies are not identical to the other [9]. The hierarchical interconnections between *AS*es bring out different types of networks [10]; (i) ***Tier-1*** *Network*: A network that can produce inbound/outbound traffic without any monetary charge regardless of the amount, direction, or type of traffic, (ii) ***Tier-2*** *Network*: A network that requires to purchase *IP* transit or pay settlements to reach at least some portion of the *Internet*, (iii) ***Tier-3*** *Network*: A network that needs to purchase transit to participate in the *Internet*, (iv) ***Internet Exchange Point*** **(*IXP*)**: A network through which networks (e.g., *ISP*s) directly exchange the *Internet* traffic between their *AS*es typically with unlimited data transfer and free of charge. However, it is difficult to practically determine the level of tiers because the business agreements between *AS*es can be private. Researchers [11] discusses the insight into the operational developments, explanation of complexity, nature of bargaining processes, implications for cost/revenue allocation and incentives, et cetera, regarding interconnections between *AS*es.

1.2.3 The *BGP* Finite State Machine (*BGP* FSM)

*BGP Speaker*s can be represented by one of six states in operation to communicate with peers [1]. Figure 1-3 illustrates the transition between the states, which are *Idle*, *Connect*, *Active*, *OpenSent*, *OpenConfirm*, and *Established*. A *BGP Speaker* in the ***Idle*** state initiates a *TCP* connection to a peer. In the meantime, all inbound traffic is ignored. While the speaker is waiting for the completion of the connection, it takes the ***Connect*** state. The state of speaker is then changed into the ***OpenSent*** state once the *TCP* connection is established with success. In the state, a *BGP Open* message is

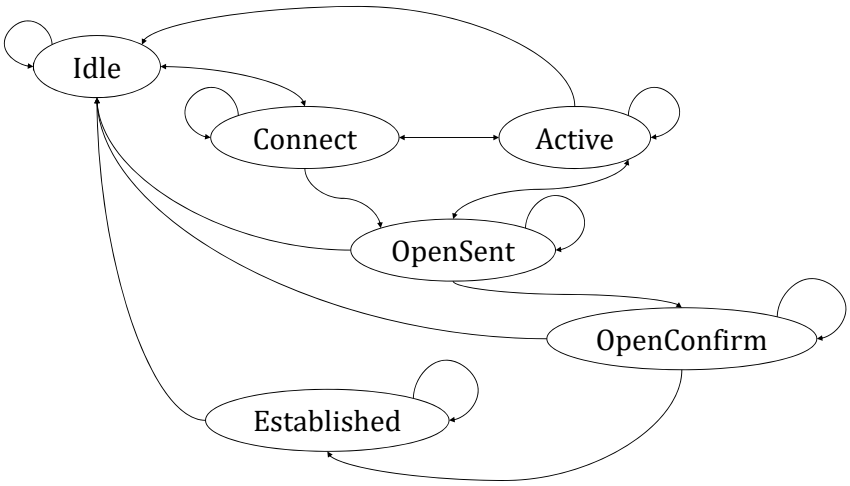


Figure 1-3: The Finite State Machine (FSM) of a *BGP Speaker*

sent out to a peer and a paired *BGP Open* message is expected to be received from the peer. The failure of *TCP* connection in the *Connect* state results in the transition to the ***Active*** state. A *BGP Keepalive* message is sent out in the ***OpenConfirm***, and a coupled *BGP Keepalive* message is anticipated as a response from the peer who has received the outgoing *BGP Keepalive* message. Upon the successful exchange of the *BGP Open* messages and *BGP Keepalive* messages, a *BGP Speaker* reaches the ***Established*** state and exchanges *BGP* messages such as *BGP Update*, *BGP Keepalive*, or *BGP Notification* messages.

1.2.4 The *BGP* Messages

The *BGP* facilitates the *Transmission Control Protocol* (*TCP*) to delegate all functionalities of the *Host-to-Host Transport (a.k.a., Transport)* layer in the *TCP/IP* model. In other words, a *BGP* message is packetized into the *Data* field in a *TCP* message segment for the announcement to *BGP* neighbors. There are four types of messages in *BGP* [1]: *BGP Open*, *BGP Update*, *BGP Keepalive*, and *BGP Notification* messages. ***BGP Open*** messages are used to establish **a *BGP* session** between two directly-connected *BGP* speakers. Reachability information is advertised between *BGP* speakers in

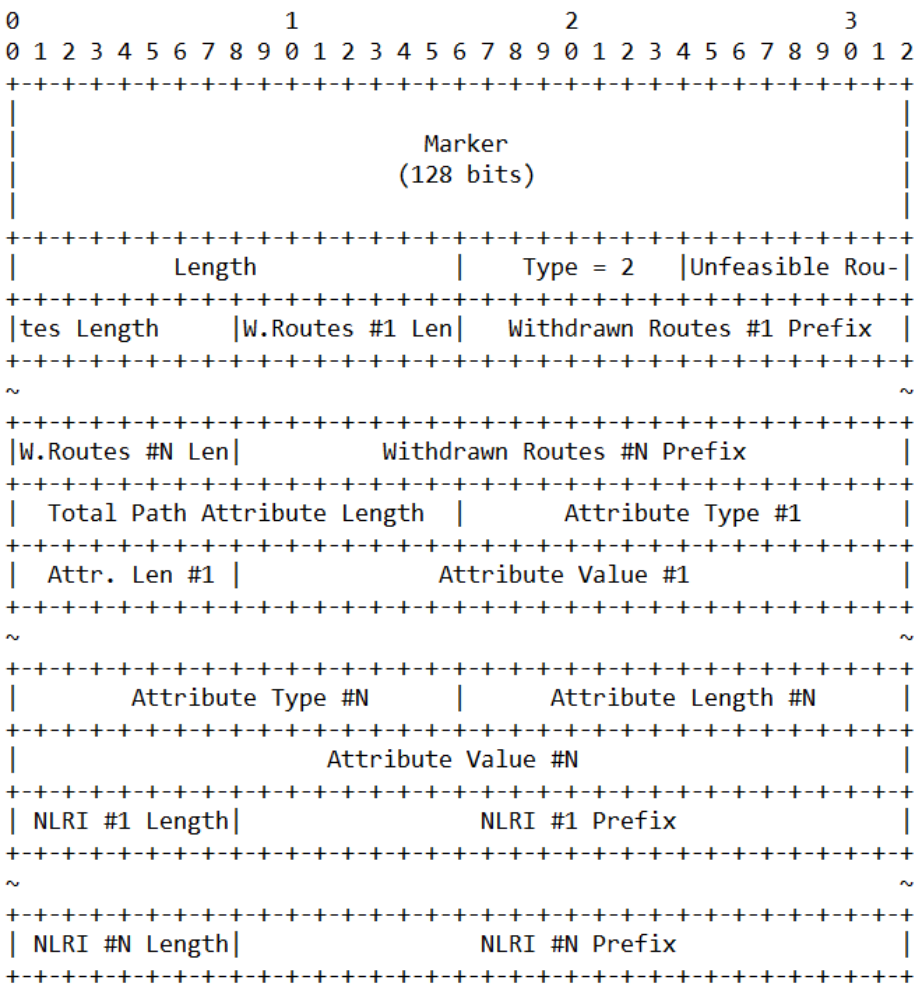
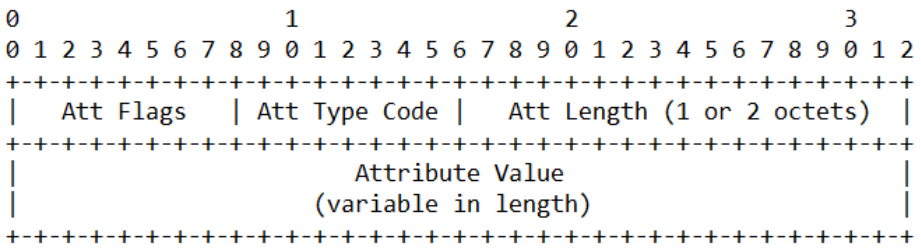


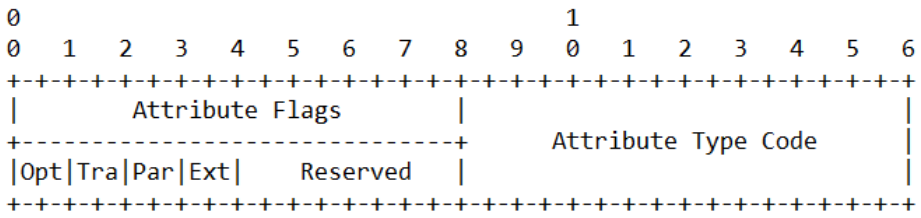
Figure 1-4: The Packet Format of a *BGP Update* Message [1]

BGP Update messages. The connectivity of a session is ensured by ***BGP Keepalive*** messages. ***BGP Notification*** messages are leveraged to notify users of an error condition when a session is unavailable.

Speaking of the format of a *BGP Update* message [1], it roughly consists of the *BGP* header, *Withdrawn Routes*, and announcements (i.e., *Path Attributes*) in the *Data* field of a *TCP* packet, as shown in Figure 1-4 in octets. A *BGP* route is defined and propagated in a *BGP Update* message as a unit of information that pairs a set of destinations (i.e., the *Network Layer Reachability Information* (*NLRI*)) with the attributes of a path (i.e., *Path Attributes*) to the destinations. In particular, the set of destinations, each of which is an *IP* address prefix consisting of successive *IP* addresses,



(a) The Format of *Attribute Type* [1]



(b) The Format of *Attribute Flags* in the *Attribute Type* [1]

Figure 1-5: The Details of *Attribute Type*s in the *Path Attribute* field of a *BGP Update* Message

is carried in one or more *NLRI* fields of a *BGP Update* message, and the set of ***Autonomous System Number*s (*ASN*s)** in a sequence is documented in the *Path Attribute* field of the same *BGP Update* message, as a route to the destinations. On the other hand, a *BGP* route can be withdrawn by (i) advertising the same route in the *Withdrawn Routes* field of a *BGP Update* message, (ii) announcing a replacement route in the *Path Attribute* field of a *BGP Update* message, (iii) or disconnecting an associated *BGP* session in such a way that all routes including the session will be removed.

Each *Path Attribute* follows the *Attribute Type* format described in Figure 1-5a and the type of an attribute is determined by the *Attribute Type Code* field [1]. Whether the attribute should be processed is defined in the *Attribute Flags* field as in Figure 1-5b. In doing so, the *Path Attribute* of a *BGP Update* message is basically used to identify eight types of *BGP* attributes. The other types of *Path Attribute*s [12] are out of scope in most cases. Note that the ***Well-known*** attributes must be supported by all *BGP* implementations; on the other hand, *BGP* implementations may support the ***Optional*** attributes. In addition, the ***Mandatory*** attributes must be included within every route

Table 1.1: The Classifications of *Path Attributes* in Processing

Attribute Type	***Path Attribute***
Well-known Mandatory	*Origin*, *AS_ Path*, *Next-hop*
Well-known Discretionary	*Local_ Pref*, *Atomic_ Aggregate*
Optional Transitive	*Aggregator*, *Communities*
Optional Non-transitive	*MED*

entry whereas the ***Discretionary*** attributes are not required to be. Among the eight *Path Attributes* that are classified by necessity as shown in Table 1.1, (i) the ***Origin***, which is *Well-known Mandatory*, informs all *AS*es how the prefix introduced into *BGP*; that is, it will be represented as *IGP*, *EGP*, or *incomplete*. (ii) the ***AS_ Path***, which is another *Well-known Mandatory*, manages a sequence of *ASN*s that a route has traversed and is intuitively used to detect loops and to build path metrics in selecting the best path. (iii) The last *Well-known Mandatory*, which is ***Next-hop*** attribute, provides the next hop *IP* address in order to reach a destination. In specific, it specifies the *IP* address of an *eBGP* neighbor in *eBGP*, as well as in *iBGP* [13]. (iv) The ***Local_ Pref*** attribute, which is *Well-known Discretionary*, is used to determine the preferred border router within an *AS* for outbound traffic; that is, the use of this attribute is limited into *iBGP* neighbors. (v) Conversely, the preferable border router of an *AS* for incoming traffic is decided by the ***MED*** attribute. The *MED* value, which is *Optional Non-transitive*, is propagated to only *eBGP* neighbors. (vi) The ***Atomic_ Aggregate*** attribute, which is *Well-known Discretionary*, is used to aggregate routes to less specific routes; in other words, the less specific routes are advertised to *BGP* peers. Practically, using the keyword "*summary-only*" results in the advertisement of only aggregate address, excluding more specific prefixes. (vii) The ***Aggregator*** attribute, which is *Optional Transitive*, provides additional information on where the aggregation using the *Atomic_ Aggregate* attribute was performed. In specific, this attribute includes the *ASN* and *IP* address of a router that

initiated the aggregate route. Note that *Cisco* facilitates the *Router ID* (*RID*) as the address of *Aggregator*. (viii) The ***Communities*** attribute [14], which is *Optional Transitive*, commonly forms four octets of *a local ASN*:*flags* (e.g., 10000:13524) and is utilized to share a common policy across multiple *BGP* peers in the same group. For example, an *AS* can set a *Communities* attribute for some of its *BGP* peers to assign the *Local_Pref* and *MED* attributes for the community, rather than individually setting these values for each of the peers.

1.2.5 The Routing Table of a *BGP* speaker

BGP Update messages play a central role to exchange routing information in *BGP* between *BGP Speaker*s. In this function, the information is stored in a special type of database, which is called the ***Routing Information Base* (*RIB*)**. The *RIB* is comprised of three components; that is, *Adj-Ribs-In*, *Loc-RIB*, and *Adj-Ribs-Out*. Speaking of the *RIB* in *BGP*, (i) the ***Adj-Ribs-In*** stores routing information learned from inbound *BGP Update* messages that were received from other *BGP* speakers, (ii) the ***Loc-RIB*** contains the local routing information that the *BGP* speaker selected, by applying its local policies to the routing information contained in its *Adj-Ribs-In*, (iii) and the ***Adj-Ribs-Out*** maintains information the local *BGP* speaker selected for advertisement to its peers. The routing information stored in the *Adj-Ribs-Out* is used to organize the routes for advertisement to specific peers by means of the local speaker's *BGP Update* messages [15] when the information is allowed by local outbound policies.

Every routing protocol technically owns its *RIB* (e.g., *BGP Adj-Ribs-In* or *OSPF Adj-Ribs-In*), all of which are integrated together to make the main routing table called the *Loc-RIB*. The *RIB* is basically operated in a similar fashion with the *Address Resolution Protocol* (*ARP*), which is used to resolve the *Network Layer* addresses into the *Data-Link Layer* addresses in the *OSI* model. The *ARP* table is built in software and is copied to the *Adjacency table*,

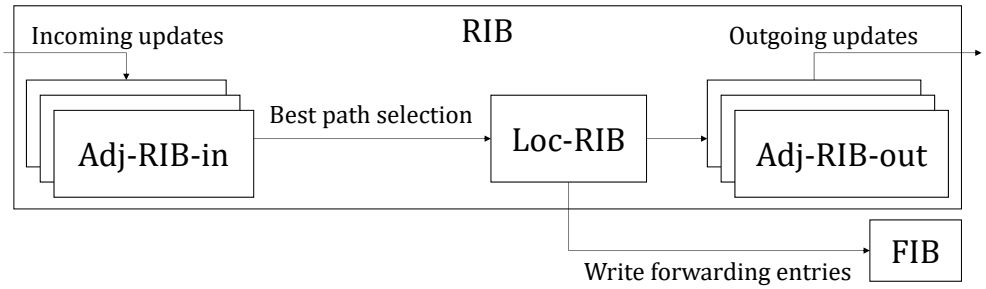


Figure 1-6: The High-level Visualization of *Routing Information Base* (*RIB*)

which is hardware. Likewise, the *RIB* entries in software are used to construct the ***Forwarding Information Base* (*FIB*)** in hardware. All information required to actually forward *IP* packets (e.g., L2 reachability information for the *Next-hop* addresses in *RIB*) are specified and maintained by the *FIB*. The router component on which *RIB* is updated in entries is referred to as the ***Control Plane***. In other words, *IP* packets are consumed and processed by a router on this plane. When packets go "through" a router, they stop by the ***Data Plane*** (a.k.a. *Forwarding Plane*) which is used to forward traffic to the next hop heading the selected destination along a path; in this case, locally used routes are not updated so, intuitively, no communication between protocols is produced in order to update a routing table.

1.2.6 The Decision Factors in *BGP* Routing

Within a *BGP* speaker, the best path is decided by leveraging various aspects [15]. This process is referred to as the ***Best Path Selection Algorithm* (*BPSA*)** in this paper. The criteria that affect the *BPSA* are (i) the monetary relationship between *AS*es in terms of allocating *IP Prefix*es, (ii) local *BGP* policies, (iii) and tie-breaking rules.

Business Relationship in *BGP*

In order for a network to reach any specific other network on the *Internet*, it must either (i) sell *transit* (or the *Internet* access) service to any other

network, making them a *customer*, (ii) peer directly with a network, which may sell *transit* service to the network, (iii) pay a network for *transit* access. The *Internet* is based on the principle of global reachability (sometimes called end-to-end reachability), which means that any *Internet* user can reach any other *Internet* user as though they located on the same network. Subsequently, any *Internet*-connected network must by definition either pay or peer with a network for *transit*. In summary, all *BGP* sessions can be categorized into (i) the ***Transit* (or *pay*)** relationship upon which a network operator pays money (or settlement) to another network (i.e., *ISP*) for the *Internet* access (or transit), (ii) the ***Peer* (or *swap*)** relationship under which two networks openly exchange their traffic for mutual benefits, (iii) and the ***Customer* (or *sell*)** relationship under which a network pays another network to be provided with the *Internet* access. Note that when two networks are in a peering relationship and belong to the same organization, the relationship can be referred to as the ***Sibling*** relationship.

Wang and Gao [16] found that routing preference conforms to *AS* relationships in most *AS*es. Namely, routes learned from customers are typically preferred over those from providers and peers, and routes received from peers are typically preferred over those from providers. Their study also revealed that the main reason for the selective announcement (i.e., customer *AS*es announce their *IP Prefix*es to a subset of provider *AS*es) is due to the load balancing, rather than the splitting/aggregating of *IP Prefix*es. Figure 1-7 shows an example of the *BGP* preference between different *BGP* sessions; that is, the business relationships between the routers. Note that all *BGP* sessions and speakers in Figure 1-7 are considered legitimate ones. An unidirectional link in the figure refers to a *Customer-to-Provider* link (i.e., a pointed router is a provider) while a bidirectional link refers to a *Peer-to-Peer* link. In other words, $AS1$ and $AS2$, which are in a higher tier than $AS3$ and $AS4$, provide *Internet* service to their customers; to be specific, $AS2$ has two customers (i.e., $AS3$ and $AS4$), and $AS3$ is a multi-homed *AS* served by two *ISP*s, which are

$AS1$ and $AS2$, for *Internet* traffic. Irrespective of the criteria in the *BPSA*, packets will not traverse some paths. For example, $AS1$ cannot send a packet to AS4 through the path $AS1$-$AS3$-$AS4$. This path is not consistent with the convergence-guaranteed conditions specified in the ***Gao-Rexford model* (*GR model*)**. Speaking of the conditions, Gao and Rexford [17] state in the *GR model* (i) that a *BGP* speaker prefers a customer route (i.e., a "down-hill" path) to a peer route (i.e., a "flat" path]) to a provider route (i.e., an "uphill" path), (ii) and that routes learned from a peer or provider cannot be exported to another peer or provider. In addition, the requirements for *GR model* can be referred to as "Valley-free;" that is, once a routing advertisement traverses a *Provider-to-Customer* edge or a *Peer-to-Peer* edge, the advertisement cannot traverse another *Peer-to-Peer* or *Customer-to-Provider* edge. Wang and Gao [16] show that a large majority of ASes on the *Internet* tend to assign higher *Local_Pref* values to customer routes than to peer routes than to provider routes. In other words, the expected path in the example is $AS1$-$AS2$-$AS4$ for forwarding packets from $AS1$ to $AS4$. The path $AS3$-$AS1$-$AS2$-$AS4$ demonstrates another example which does not agree with the *GR model*. Intuitively from the figure, three paths are available to send a packet to $AS4$ on the perspective of $AS3$; that is, $AS3$-$AS1$-$AS2$-$AS4$, $AS3$-$AS2$-$AS4$, and $AS3$-$AS4$. The $AS3$ prefers the path $AS3$-$AS4$ instead of facilitating one of the upstream paths (i.e., through its provider $AS1$ or $AS2$). Unselected paths may be dropped or leveraged as a backup path; the decision literally depends on the local policies of a *BGP* speaker.

BGP Policies

A *BGP* speaker consults its inbound/outbound policies discussed in Section 1.2.5, as well as the *BPSA* mentioned in Section 1.2.6, in order to generate a whitelist in terms of routing. More studies regarding the policies are considered in this section. Caesar and Rexford [18] discuss common patterns of designing policies. Each *BGP* speaker has its own policy to add or remove

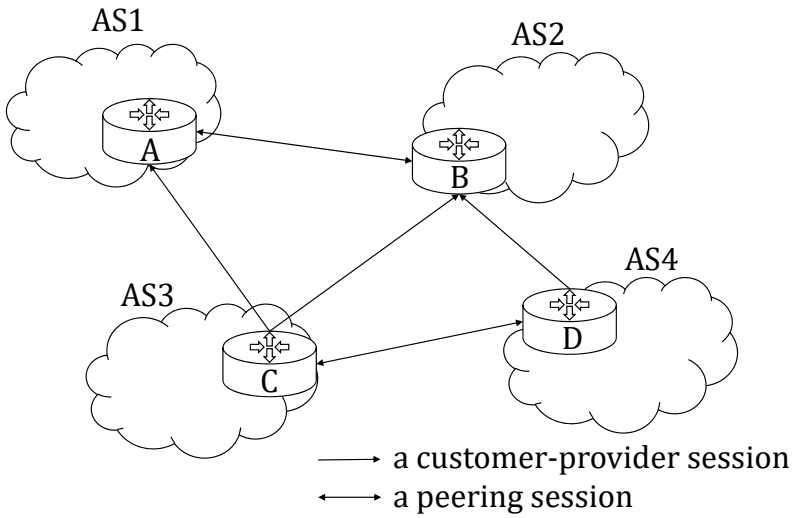


Figure 1-7: The Example of *BGP* Paths Determined by Business Relationships Implying the Gao-Rexford Model

BGP Update messages received from its peers. In other words, *AS*es can choose their policies on their own so the absence of conflicting policies in inter-domain routing cannot be guaranteed [19]. Similarly, *Neighbor Based Import and Export (NBIE)* phenomenon exists, which means that an *AS* usually does not differentiate *AS*es which are non-adjacent to it [20]. Gao and Rexford [17] propose a set of guidelines for an *AS* to follow in setting its routing policies, without requiring coordination with other *AS*es. The guideline provides each *AS* with significant flexibility in selecting its local policies and ensures route convergence even under changes in the topology and routing policies.

A stub *AS* refers to an *AS* that is connected to only one other *AS* and sends out much or all of its non-local traffic through the single path. If a stub *AS* is an access provider, it usually has much more inbound traffic than outbound traffic. In this case, it often needs to control its incoming traffic rather than controlling outbound traffic by applying its tie-breaking rules, and the situation can be complicated. Indeed, the stub domain needs to influence the decisions to the other routers in other domains (i.e., its customer *AS*es in this case). For the issue, the *AS-Path Prepending* or *BGP Communities* [21] can be leveraged to control incoming traffic. That is to

say, a *BGP* speaker influences the paths for incoming traffic by facilitating pre-agreed *Community* values to inform other *AS*es about the *Local_Pref*, artificial increase of *AS_Path* length to make the path less attractive, usage of inter-*AS* metrics (i.e., *MED*), et cetera.

Tie-breaking Rules

When multiple routes are available in the *Adj-Ribs-In* to the same destination, a *BGP* speaker must prioritize those routes and select one as the best path via tie-breaking rules so as to update its *Loc-RIB* with the preferred path. This process is so called *BPSA*, in which many rules are provided to determine the best path [15]. Based on the *BPSA* of *Cisco* routers [15], the following list enumerates all attributes [1] that are utilized in the algorithm in order of preference; that is, the first one is most preferred while the last is least, and the order is leveraged by most border routers. A *BGP* speaker prefers a path (i) that owns the **highest *Local_Pref* value**, (ii) that **locally originated** via a network or aggregate *BGP* subcommand or through redistribution from *IGP*, (iii) that possesses the **shortest *AS_Path***, (iv) that shows the **lowest origin type** (i.e., *IGP*<*EGP*<*incomplete*), (v) that has the **lowest *MED*** value, (vi) that is **an *eBGP* path** instead of an *iBGP* one, (vii) that holds the **lowest *IGP* metric** to the *BGP* next hop, (viii) that was received earlier in a timely fashion (i.e., **the oldest one**), (ix) that comes from a *BGP* speaker with the **lowest router ID**, (x) that is with the **minimum cluster list length**, (xi) that provides the **lowest neighbor address**. Actual *BPSA* can differ router-by-router, and the common design patterns of *BPSA* employed by *ISP*s were observed by Caesar and Rexford [18]. Note that, for the purpose of load balancing in traffic, a *BGP* speaker can identify multiple *BGP* paths in the *Loc-RIB* to the same destination by leveraging the *BGP Multipath*. Most likely, the *Local_Pref* value is used to install local policies; for example, a higher *Local_Pref* value will be assigned to in-policy acceptable routes than to unsatisfactory ones.

Up until this point, the fundamentals of *BGP* discussed in Section 1.2 covers only topics that are highly correlated with our studies. Further information can be provided by the document [1] that is published by the *Internet Engineering Task Force* (*IETF*) and specifies the *BGP* standards. We focused on validating *BGP* origins and paths by providing corresponding capabilities so as to protect *BGP* from all *BGP*-originating vulnerabilities. In terms of cryptography, *BGP* does not offer the *Confidentiality*, *Integrity*, and *Authenticity*. Because *BGP* is operated over *TCP*, *Confidentiality* requires being supported by *TCP*, rather than *BGP*. In other words, our study is concentrated on providing the *Origin Validation* and *Path Validation* into *BGP* for the *Integrity* and *Authenticity*. Our approach to secure *BGP* is to (i) find and select a solution for the *Origin Validation* and *Path Validation*, (ii) construct an experimental network based on the chosen solution, (iii) simulate the network, (iv) and, if any problem is observed, enhance the security level or performance of the network. Correspondingly, this paper is organized as follows. Section 2 demonstrates the related literature regarding *BGP*. Section 3 elaborates our implementations, simulations, as well as enhancements based on *Resource Public Key Infrastructure* (*RPKI*) [22]. The *BGPsec* [2], which is the most remarkable solution in validating *BGP* paths, is re-considered in its signature algorithm for better performance in Section 4. Moreover, our comprehensive *BGP* data-analyzing tool is introduced in Section 5, using which a few algorithms and their simulation results to reduce the number of paths to be validated are uncovered in Section 6. Our studies are summarized and concluded in Section 7 to put a period.

Chapter 2

Literature Reviews

The *BGP* [1], specifically *BGP*-4, is originally designed to operate in a trusted environment, and there are no internal mechanisms to protect the information it carries [23]. Intuitively, it is exposed to innumerable vulnerabilities that cannot be easily detected and rectified due to the complexity of *BGP*, in such a way that severe accidents occurred and injured *BGP* networks. While some accidents have limited impact and scope, others may lead to significant and widespread damage as discussed in Section 2.2.

2.1 The *BGP* Problems

In general, the *BGP* depends on hearsay since each *BGP* speaker believes and repeats what it has heard from its neighbors. By that means, most accidents occurred because of the wrong *Network Layer Reachability Information* (*NLRI*) or *Autonomous System* (*AS*) path. The ***NLRI*** carries the one or more *IP Prefix*es that an origin *AS* is authorized to announce. The ***AS* path** is a sequential list of *Autonomous System Number*s (*ASN*s) all the way to a destination from the origin *AS*. In other words, if any *IP* packet including an inaccurately provided *NLRI* or *AS* path will be facilitated to generate a malicious route, bring out a *BGP* accident such as a route leakage or hijacking.

As few more remarks by definition in *BGP*, all vulnerabilities that

originated from *TCP* are inherited into *BGP* because all error control functions are delegated to *TCP* by operating *BGP* over *TCP*; this idea is to make *BGP* much simpler [24]. Speaking of *TCP*, the original proposal for *TCP*-level message integrity, which is *TCP Message-Digest algorithm 5* (*MD5*) [25], is considered cryptographically broken [26]. Furthermore, *TCP MD5* does not provide *per-packet authentication*, *integrity*, *confidentiality*, or *replay protection*. The *National Institute of Standards and Technology* (*NIST*) encourages application/protocol designers to implement *Secure Hash Algorithm*-256 (*SHA*-256) at a minimum for any application of hash functions, rather than the vulnerable *MD5* [27]. Anyway, *TCP*-originated problems are not concerned in this paper so as to make our study concentrate on *BGP* itself. In addition, it is important to remember that *BGP* is a routing protocol that operates at an inter-*AS* level. That is to say, routes are chosen between *AS*es and not at the level of individual routers within an *AS*; specifically, when *BGP* stores information of a path to a network, the information is stored as a sequence of *ASN*s not as that of specific routers. Namely, *BGP* cannot deal with individual routers in an *AS* because by definition, resulting in an additional deficiency. Generic threat information for routing protocols [28] and particular *BGP* security requirements [29] are also available.

Mahajan et al. [30] reveal from their quantitative study that configuration errors are pervasive, and 200 to 1200 prefixes (0.2 to one percent of the *BGP* table in size) are suffering from misconfiguration every day. They also found that connectivity is robust to most misconfigurations. Only four percent of the misconfigured announcements or 13 percent of the misconfiguration incidents affect the connectivity. On the other hand, the routing load originating from the misconfiguration is significant. Although most misconfiguration is the largest contributor to malicious announcements, they believe that most of these faults cannot be prevented through a better router design because the causes of misconfiguration are diverse. Boothe et al. [31] found that router user interfaces are a much larger threat to the *BGP*-layer of the

Internet than malicious operators. In specific, out of approximately 200, 000 unique (*AS*, prefix) pairs in June 2006, 90 and 4000 pairs were considered malicious and erroneous originating from misconfiguration. Vissicchio et al. [32] show that routing and forwarding anomalies can originate from *BGP* misconfiguration, even when *MED* is not used and simple policies are accepted. Subsequently, they propose a solution that leverages multiple isolated control planes in parallel for lossless *BGP* reconfiguration. Bellovin and Gansner [33] demonstrate a link-cutting attack and provided the feasible calculations to effectively cut links. Lad et al. [34] show that *BGP* allows local connectivity dynamics to be propagated over the globe. Note that the ***BGP dynamics*** mean forces that stimulate a change or progress within the *BGP*. As a result, any small number of edge networks can potentially cause wide-scale routing overload. They also revealed in their simulation that the *BGP Route Flap Dampening* (*RFD*) proposed to solve this problem had not been fully deployed and such a partial deployment of dampening not only limits effect but may also worsen the routing performance under certain topological conditions. Jintae Kim et al. [35] shows that there are many instances in the *AS*-level topology where their methods can successfully attack a victim. Ramachandran and Feamster [36] found that most spams are being sent from a few regions of *IP* address spaces, and that spammers appear to be using transient "bots" that send only a few pieces of email over very short periods of time. Finally, a small, yet non-negligible, amount of spams are received from *IP* addresses that correspond to short-lived *BGP* routes, typically included in hijacked prefixes. These trends suggest that developing algorithms to identify botnet membership, filtering email messages based on network-level properties (which are less variable than email contents), and improving the security of the *Internet* routing infrastructure, may prove to be extremely effective in combating spam. Ballani et al. [37] studied prefix interception and revealed that *AS*es higher up in the routing hierarchy can both hijack and intercept traffic to any prefix from a significant fraction (i.e., larger than

50 percent) of *AS*es in the *Internet*. Small *AS*es can hijack and intercept traffic from a non-negligible fraction of *AS*es. Pilosov and Kapela [38] found that any arbitrary prefix can be hijacked, without breaking end-to-end (i.e., the *Man-In-The-Middle* (*MITM*) attack). The demonstration at the largest annual hacker convention [39] manifested the possibility of using *BGP* to isolate a designated network or manipulate traffic maliciously by hackers. Montgomery and Murphy [40] discuss that the protocols, data, and algorithms that compute paths through interconnected network devices are possibly the most vital, complex, and fragile components in the global information infrastructure, and they are the least protected. S. Murphy [23] considers some security issues in disseminating *BGP* routing data. Zheng et al. [41] develop a scheme to detect *IP Prefix Hijacking* in a real-time fashion, using two key observations which are the hop-count stability and *AS* path similarity. They also argued the accuracy of their contribution because most *AS* paths are stable. In addition, the similarity should be preserved as far as the geographical location of an *AS* is not changed. Zhang et al. [42] present a *IP Prefix Hijacking* detection scheme by monitoring network reachability from key external transit networks to one's own network through lightweight prefix-owner-based active probing. However, the accuracy of their base data derived from *traceroute* can be questioned as discussed in [9]. Leavitt [43] estimates that the projected *IPv4*-address exhaustion date will be February 2012 for European networks; July 25 2013 for African networks; December 17 2013 for the *US*, Canada, and some Caribbean islands; and April 9 2014 for Latin America and other parts of the Caribbean.

2.2 The Past Accidents in *BGP*

On April 25 1997, a misconfigured router maintained by a small service provider in Florida injected incorrect routing information into the global *Internet* and claimed to have optimal connectivity to all *Internet* destinations.

Because such statements were not validated in any way, they were widely accepted. As a result, most *Internet* traffic was routed to this small *ISP*. The traffic overwhelmed the misconfigured and intermediate routers, and effectively crippled the *Internet* for almost two hours [44]. In the same year, *AS*7007 [45] accidentally advertised to its upstream provider a short path to numerous *IP Prefix*es belonging to other networks. Its provider did not filter out these bogus announcements causing a large *black-hole* for many destinations. *Google* went down around 22:10, May 7 2005 *UTC* for 15 minutes to an hour. *Google* explained about the accident that it was caused by misconfiguring internal *DNS*. However, Tao Wan and van Oorschot [46] reveal that *AS*174 operated by *Cogent* somehow originated routes for one of the *IP Prefix*es that *Google* owns (i.e., 134.233.161.0/24), immediately prior to the outage. This mis-behavior brought out that 49.1 percent of *AS*es re-advertising routes for the *compromised* prefix switched to the incorrect path. Speaking of the "***compromised***," the expression is used to represent that someone or something has maliciously broken into a router without permission so the integrity of any packet in the router cannot be guaranteed; literally, the phrase is widely accepted in this paper and in this field of study.

At 05:05:33 *UTC* on January 22 2006, *Con Edison* (a.k.a. *ConEd*) announced a number prefixes (networks) owned by their customers. The normal way of routing traffic is that providers wait for their customers to announce networks and then just repeat the announcement. In this case, *ConEd* started pretending to be its own customers. It would be probably not service-impacting, assuming that *ConEd* had known how to get the traffic to the customers, but it was odd. Todd Underwood [47] from *Dyn Research* (a.k.a *Renesys*) states that *ConEd* spewed out huge numbers of announcements for the next several minutes. One of the first customers observed by *Dyn Research* (a.k.a *Renesys*) was *Martha Stewart Living Omnimedia*. It is worth noting that this event was three and a half hours earlier than the problem that *Panix* (*AS*2033) complained about. So something strange was already at work at

ConEd. Just as in the case of real identity theft, it will be appreciated if someone does not believe those who are pretending to be you to steal your properties or belongings (in this particular case of *ConEd* the object is your networks). The incredulous was *UUnet* (*AS*701) at that time. They kept believing the first set of lies until 05:22:29 *UTC* when the networks started moving back to their rightful owners. Subsequently at 08:23:12 *UTC*, *Verio (a.k.a. NTT America)*, *AS*2914) started believing some of the same lies that *UUnet* was already believing and *ConEd* started telling more lies. At 08:31:15 *UTC*, the first instance of *ConEd* announcing *Panix*'s route appeared, and that one only ever appeared through *Verio*. The worst part of the accident was that some of these networks were still being hijacked by *ConEd* as of about 04:00 *UTC* on January 23 2006.

On February 24 2008 *UTC*, the Pakistan government sent out a most urgent corrigendum to the *Pakistan Telecommunication Authority* (a.k.a. *PTA*, *AS*17557) to block an offensive website provided by *Youtube* (*AS*36561). In order to have the website blocked in Pakistan, although the *PTA* was supposed to drop the related *IP* packets, it started the unauthorized announcement of prefix 208.65.153.0/24 that was more specific than the prefix owned by *Youtube* (i.e., 208.65.153.0/22). This conscious or unconscious configuration of *PTA* took place at 20:07 February 24 2008 *UTC*. One of *PTA*'s upstream providers, *PCCW Global (AS3491)* forwarded this announcement to the rest of the *Internet*, which resulted in the hijacking of *Youtube* traffic on a global scale during almost two hours [48]. Similarly, Zmijewski [49] reveals that two *US* DoD networks were hijacked at least seven times.

The Hengchun earthquakes occured on December 26 2008 at 12:26 and 12:34 *UTC* near the southwest coast of Taiwan damaged seven of the eight cables connected to Taiwan, disrupting communications in much of Asia. Cable operators required several months to have the injured service entirely restored. According to the Telegraphy [50], on the morning of January 30 2008, two international submarine cables, which are the *FLAG Euripe-Asia*

cable operated by *FLAG Telecom* and *SeaMeWe-4 (South East Asia-Middle East-Western Europe-4)* that is a consortium cable owned jointly by fifteen telecommunication companies, in the Mediterranean Sea were damaged, causing significant disruptions to the *Internet* and phone traffic dominantly between Europe and the Middle East. The cable cuts reduced the amount of available capacity on this direct route to Europe by 75 percent (620 $Gbps$).

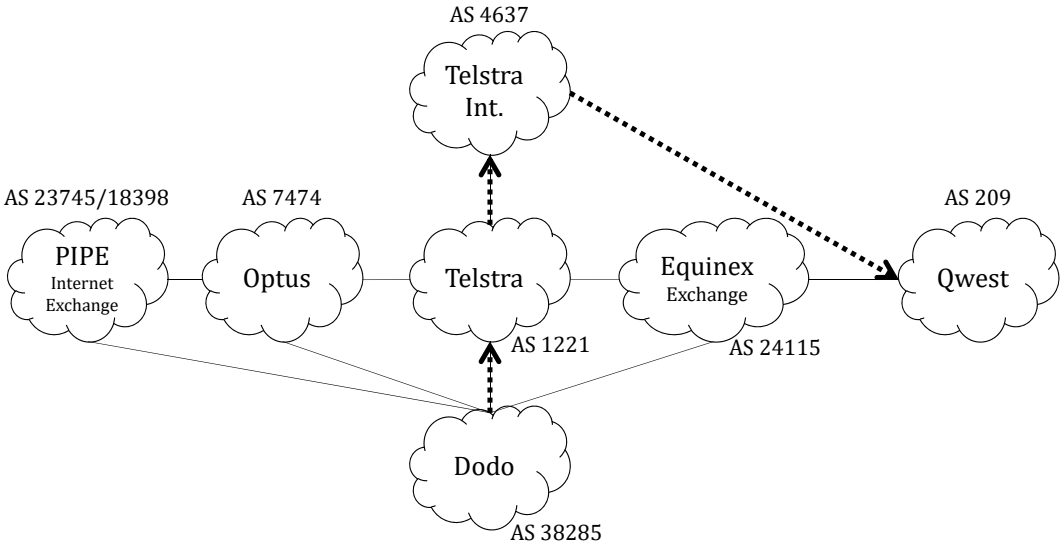
The *SuproNet*, which was a local Czech *ISP*, single-handedly caused a global *Internet* meltdown for upwards of an hour on February 16 2009, resulting in the *Internet* bedlam [51]. Speaking of which, the AS path propagated by the backup provider of *SuproNet* were somehow accepted by adjacent ASes; in detail, the *Sloane Park Property Trust* ($AS29133$) was having the route to a prefix 94.125.216.0/21, which is owned by *SuproNet*, with an AS path exceeding 255 ASNs; note that most ASNs in the path were inserted by the *AS-Path Prepending* [1]. The *Level-3* ($AS3356$), *Tiscali* ($AS3257$), and *TeliaSonera* ($AS1299$) much more aggravated the situation by propagating the route to the globe. In summary, the single provider announcing a single malicious prefix caused a huge increase in the global rate of *BGP Update* messages, peaking at 107, 780 updates per second. China's government diverted 15 percent of the *Internet* traffic for 18 minutes in April 8 2010 UTC, resulting in intercepting sensitive traffic [52]. On the day, starting at 15:50, a small part of *China Telecom* ($AS23724$) incorrectly asserted ownership of more than 50, 000 different blocks of IP addresses, which occupied the 15 percent of the *Internet*. Nobody had doubts about those assertions. As a consequence, the size of their routing table ($AS23724$) grew to more than 1000 times larger than the normal size within a few minutes; intuitively, the AS started to receive traffic from 50, 000 networks. At around 02:24 UTC on November 6 2012, *Cloudflare* noticed that *Google*'s services were offline [53]. The accident is evidenced by that the BGP routes for a *Google* ($AS15169$) IP address on *Cloudflare* ($AS13335$) traversed *Moratel* ($AS23947$), which is an Indonesian *ISP*, rather than directly headed to $AS15169$. Thereby, *Google*

experienced a limited outage for about 27 minutes over some portions of the *Internet*.

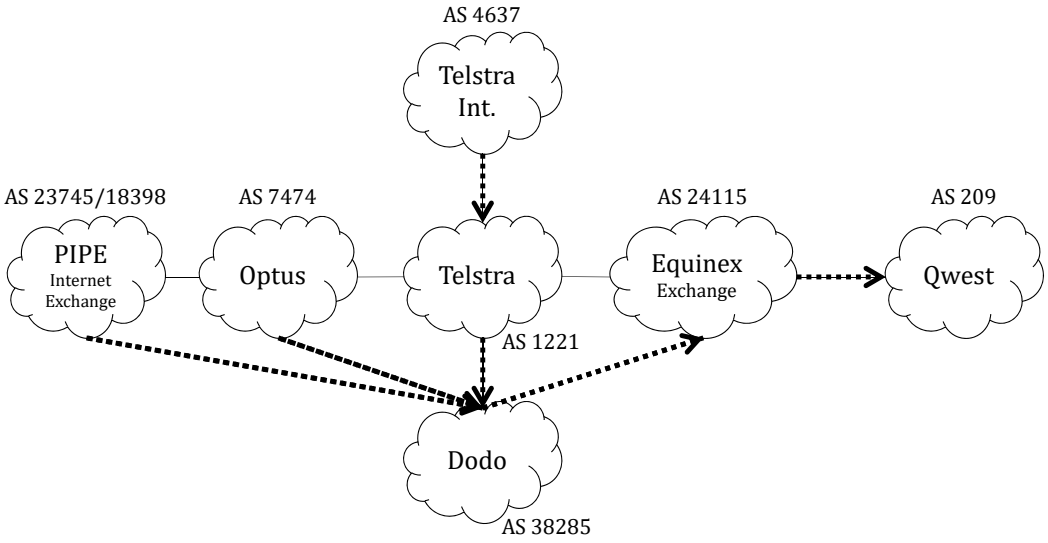
Figure 2-1 illustrates the accident happened in Australia on February 23 2012 [54]. A normal path was 4608-1221-*4637*-209-20940-20940 as shown in Figure 2-1a. At 2:39 am *UTC*, when *Telstra* received an announcement of the *AS_Path* 4608-1221-**38285-24115**-209-20940-20940, they should drop the announcement. However, the newly learned path actually originated from *Dodo*, which was one of customer *AS*es of *Telstra*. In other words, the path was selected as the best path by *Telstra* corresponding to its local policy stating that a customer route is preferred to a provider route. Similar situation is grabbed by *Optus*, as well as by *PIPE Internet Exchange*, resulting in the propagation of a route to *Qwest* "through" *Dodo* as depicted in Figure 2-1b. At the end of this malicious circulation, *Dodo* received large amounts of traffic that it would normally accept. Consequently, most of traffic were out of capacity of *Dodo* and dropped by the network. UA Network Research Lab [55] developed a detection system for the *Large Route Leak*s (*LRL*s) events at real time and without requiring authoritative prefix ownership information. Based on their detection algorithm, they identified five to twenty *LRL*s by 2010 on a yearly basis from the *BGP Update* messages collected by the *RouteViews*.

2.3 The *BGP* Proposals on Design and Infrastructure

As a summary of Section 2.1 and Section 2.2, *BGP* is a vital, complex and fragile component in the global information infrastructure but relatively receives little protection [40]. Due to the trusted environment in which *BGP* is initially designed, the protocol cannot be protected from a variety of malicious attacks. For instance, *BGP* sessions are subject to wiretapping attacks. *BGP Speaker*s can be *compromised*, resulting in *Prefix Hijacking*, *black-hole* attacks,



(a) Normal Route from *Telstra* to *Qwest*



(b) Changed Route Maliciously Selected by a Local Policy of *Telstra*

Figure 2-1: The Visualization of *BGP* Accident Originating from *Telstra* on February 23, 2012

and so on. These attacks also accompany the replay/modification/suppression of valid *BGP* messages or transmission of fictitious messages. The easiest and obvious way to make *BGP* secure is to ensure proper configurations [56]. However, this is technically difficult because of the following reasons: (i) Providers do not always know which address blocks their customers are assigned to, due to the prevalence of ***Multi-homing***, which allows customers

to obtain address prefixes from multiple providers. (ii) Similar to ingress filtering, as long as there is one provider that does not properly enforce *Route-filtering*, *IP Prefix Hijacking* becomes possible. Third parties can misconfigure or manipulate a remote router; for example, by leveraging the *Communities* attribute. (iii) Most *compromised* routers in the *Internet* can bypass *BGP* filters since the *Route-filtering* is impossible along peering edges. It is due to the lack of information on the *IP Prefix*es that are allocated to networks (i.e., *AS*es) belonging to one's competitor for revenue. In these situations, serious routing problems or economic loss possibly takes place [30].

Perlman [57] shows that routers are basically so byzantine that a secure inter-domain (i.e., inter-*AS*) routing protocol must have byzantine robustness: (i) ***Termination***: In the face of malicious or faulty behavior from other hosts, all non-faulty hosts in the system should reach a decision on a particular message's contents within a finite period. (ii) ***Agreement***: The decision should be the same between all non-faulty hosts. (iii) ***Validity***: The message should be the one sent by the source node. The byzantine robustness can be achieved by providing the *Time-To-Live* (*TTL*), the unified protocol, and the required validations. We associate the *Termination* and *Agreement* in the byzantine robustness with the design or infrastructure of *BGP*, regarding which many studies have been introduced to resolve the related problems. This section is purposed to demonstrate those researches. Note that the *Validity* studies are discussed in Section 3.1. We surveyed 40 associated papers and categorized them into four topics; that is, *routing*, *traffic-engineering*, *network topology*, and *geography*-related approaches. More than 20 papers discuss the problems and solutions on *BGP* routing and concentrate on the *Control Plane*, *Data Plane*, and convergence of a *BGP* router.

2.3.1 The Studies of *BGP*-routing

Flavel et al. [58] develop a mathematical model to capture *BGP* table fluctuations in order to provide the necessary foundations to study short-

and long-term routing table growth. Kushman et al. [59] develop *R-BGP* and showed in their simulation that *R-BGP* reduces the number of domains having transient dis-connectivity originating from a link failure to zero. Akella et al. [60] show that *Multi-homing* to three or more *ISP*s and scheduling traffic across the *ISP*s can improve the *Internet Round-Trip Time*s (*RTT*s) and throughputs respectively by up to 25 and 20 percent. Park et al. [61] discover that about 13 percent of all *BGP* routing updates are duplicates and that routers can receive up to 86.4 percent of duplicates during their busiest periods, possibly leading to a negative impact on the routing load of *BGP* speakers.

In order to improve the functionalities on the *Control Plane* of a router, Bolla and Bruschi [62] develop the *Distributed software ROuter Project (DROP)*, which allows aggregating multiple *Software Router*s in a single logical *IP* node. The *DROP* is partially based on the main guidelines of the *IETF ForCES* standard, and it is specifically designed to naturally interface with Linux operating system and applications in such a way that it allows realizing a distributed fast-path among a number of *Software Router*s, managing their slow-path in an autonomous way and self-interfacing with *Linux* standard network applications and commands. Griffin and Wilfong [63] look into the *MED* oscillation problem and provided the analysis based on the *MED Induced Routing Anomalies (MIRA)*. Van den Schrieck et al. [64] analyze *iBGP* to provide the path diversity into the *iBGP* network in such a way that the *Multi Exit Discriminator* (*MED*) oscillation could be prevented. Quoitin et al. [65] show the limitations of *AS-Path Prepending*, as well as the difficulty of controlling the flow of the incoming traffic. Griffin et al. [66] introduce the *Stable Paths Problem*. The problem is examined under a structure called a *dispute wheel*, corresponding to the existence of which they showed that a unique solution to the problem can be developed. The *Simple Path Vector Protocol (SPVP)*, a distributed algorithm, is also proposed to solve the *Stable Paths Problem*. Feldmann et al. [67] propose a methodology for identifying

the origin of routing instabilities by examining and correlating *BGP Update* messages along three dimensions (i.e., *time*, *views*, and *prefixes*) and showed how it can be adapted to account for the complexities of *BGP*.

Oliveira et al. [68] present measurement results that identify *BGP* slow convergence events across the entire global routing table. According to their data, routers and prefixes in tier-1 *ISP*s are less observed in path exploration (i.e., shorter convergence delays) than that originated from edge *AS*es. Further, they introduced the method to infer path preference based on the path usage time. In order to estimate locality of routing events, Sapegin and Uhlig [69] design a method that classifies sets of *BGP Update* messages, called *Spike*s, into either *correlated* or *non-correlated* by comparing streams of *BGP Update* messages from multiple *Vantage Point*s (*VP*s). Note that a ***Correlated Spike*** of prefix updates (i.e., announcements or withdrawals) is to capture a case where the *BGP Update* message received from its neighbor contains the *IP Prefix* information that has ever been received. Li et al. [70] propose *stableBGP* that practically solve a general class of *BGP* instability such as route oscillations and path explorations.

By applying different *Peering Equilibrium Multi-Path (PEMP)* policies, Secci et al. [71] show that the routing cost can be decreased by roughly ten percent with the *PEMP*. They also showed that the routing stability can be significantly improved and that congestion can be practically avoidable on the peering links. In order to reduce *end-to-end* delay, packet loss rate, and throughput in average, Hsu and Shieh [72] develop a *Simple Path Diversity Algorithm (SPDA)* that allows *BGP* speakers to save multiple paths in its routing table and send out traffic to both the best and alternative path at the time of accident (e.g., a link failure of the best path). Similarly, Camacho et al. [73] propose the *BGP*-XM (*BGP*-eXtended Multipath) extension, of which the performance evaluation shows that high path diversity can be achieved even for limited deployments of the multi-path mechanism. The key is to facilitate different multipath selection algorithms, such as the K-Best Route Optimizer

(K-BESTRO) tunable selection algorithm, and their assembling algorithm. They also reveal that the extension is suitable for large deployment since it shows a low impact in *AS* path length and in routing table size. Mao et al. [74] show that *Route Flap Dampening* (*RFD*) can exacerbate the convergence time of relatively stable routes and introduce a preliminary proposal to modify the scheme of *RFD* to remove the undesired interaction of flap dampening.

Sriram et al. [75] show that the *Route Flap Dampening* (*RFD*), which is designed to alleviate *BGP* processing overload and router flapping under non-malicious scenarios, can be used by attackers to achieve a high probability of *AS-AS* and *AS*-prefix isolation by attacks conducted at a rate roughly equal to one per *Minimum Route Advertisement Interval* (*MRAI*), resulting in network-wide disruptions. Speaking of which, the ***MRAI*** determines the minimum amount of time that must elapse between an advertisement and/or withdrawal of routes to a particular destination by a *BGP* speaker to a peer; this rate-limiting procedure applies on a per-destination basis, although the value of *MRAI* is set on a per-peer basis. Note that *MRAI* timer is developed to add a level of synchronization to the routing system by limiting the number of *BGP Update* messages and to reduce the computational complexity of *BGP* convergence from $O(n!)$ to $O(n)$ [76]. The partial mitigation of attack can be achieved by using the *BGP Graceful Restart (BGP-GR)* mechanism. Ee et al. [77] propose a distributed mechanism that enforces a preference ordering by removing policy-induced oscillations when disputes causing those oscillations exist, in such a way that the conflict between policy autonomy and system stability could be resolved. Yilmaz and Matta [78] propose the *Adaptive Policy Management (APM)* to resolve policy conflicts between *AS*es.

Wang et al. [79] first quantify the impact of session failures on both the *Control Plane* and *Data Plane* using *syslog* events. They found that the major root causes are administrative session resets and link failures, each of which contributed to 46.1 and 30.4 percent of the observed session failures. Labovitz et al. [76] found that the average convergence latency for a route failure

corresponds to the length of the longest possible backup path allowed by policy and topology between two providers. Additionally, a probable explanation for the differences (while the longest non-vagabond path explored during convergence for tier-1 customer routes was nine *AS*es, tier-2 customer routes grew as long as 12 *AS*es) in the backup paths from different tiers of providers is that smaller *ISP*s typically purchase transit from multiple upstream providers. Obradovic [80] proves that the convergence-time *upper-bound* for a destination becoming unavailable is $O(n)$ while n is the number of AS nodes in the network and that the message-overhead *upper-bound* is $n \times m$ while m is the number of directed *BGP* sessions. Pei et al. [81] develop *BGP-RCN* in which each *BGP Update* message carries the information of what triggered the message. As a result, reduction in convergence time and the total number of routing changes was accomplished.

2.3.2 The Studies of *Traffic-engineering* in *BGP*

Labovitz et al. [82] state that a high level of instability was found in the global *Internet* backbone specifically between different service providers excluding the inside of a provider. In the given study of the internal routing of a regional provider, any daily or weekly frequency component was not found. Shaikh et al. [83] develop analytical models to quantify the effect of congestion on the robustness of *BGP* as a function of the traffic overload factor, queuing delays, and packet sizes. Their experiments show that the *resilience* of *BGP*, against congestion, decreases by increasing queuing and link propagation delays. Note that the "***resilience***" is an ability to provide and maintain an acceptable level of *BGP* routing capability in the face of faults and challenges to normal operation. The *BGP* does not facilitate common traffic engineering tasks, such as balancing load across multiple links to a neighboring AS or directing traffic to a different neighbor. Feamster et al. [84] propose fundamental objectives for inter-domain traffic engineering and specific guidelines for achieving these objectives within the context of *BGP*. Leighton [85] discusses about how

to enhance the *Internet* performance by dealing with Inter-network itself. Sriram et al. [86] discuss some architectural principles pertaining to mapping distribution protocols. They considered how *Egress Tunnel Routers (ETRs)* can perform aggregation of *End-point ID (EID)* address space belonging to their downstream delivery networks, in spite of migration/re-homing of some sub-prefixes to other *ETRs*. This aggregation may be useful for reducing the processing load and memory consumption associated with mapping messages.

2.3.3 The Studies of *BGP* Topology

Mao et al. [87] facilitate *BGP Beacon*, which refers to a publicly documented prefix having global visibility and a published schedule for announcements and withdrawals, to better analyze the global *BGP* dynamics in the *Internet* by injecting controlled *BGP Update* messages in the network. However, their research is limited to the *PSG Beacons* and *RouteViews* monitoring data. Z. Morley Mao et al. [9] show that the *IP*-to-*AS* mapping extracted from *BGP* routing tables is not sufficient for determining the *AS*-level forwarding paths and newly proposed a systematic way to construct accurate *IP*-to-*AS* mappings using dynamic programming and iterative improvement. Their algorithm in simulation reduces the initial mismatch ratio of 15 percent between *BGP* and *traceroute*-derived *AS* paths to five percent while changing only 2.9 percent of the assignments in the initial *IP*-to-*AS* mappings.

Lad et al. [88] develop the "Link-Rank" graphical tool that weighs the links between *AS*es by the number of routing prefixes going through each link, giving rise to capture *BGP* dynamics. Cohen and Raz [89] found that almost all missing *BGP* sessions are of the *Peer-to-Peer* type. Bonaventure et al. [90] propose a new fast-reroute technique where routers are prepared to react quickly to inter-domain link failures. Their results show that about 60 percent of mismatches originate from *IP*-address sharing between *BGP* speakers and that only about 14 percent of the mismatches are caused by the presence of *Internet Exchange Point*s (*IXP*s), siblings, or prefixes with multiple origin

*AS*es. Two public repositories they looked into also show 16 and 47 percent of false *AS* adjacencies. Augustin et al. [91] integrate an *Internet*-wide *traceroute* study designed to detect the unknown *IXP*-specific peering matrices with publicly available *traceroute*s and geographically dispersed *Vantage Point*s (*VP*s). Subsequently, about 44, 000 *IXP*-specific peering sessions were discovered; that is, nearly 18, 000 more sessions were found than before.

In opposition to them, Zhang et al. [92] conduct a systematic investigation into the potential errors of the *IP address to ASN* derived by *traceroute* using the *Longest Prefix Matching (LPM)* method. Oliveira et al. [93] assess the quality of the inferred *Internet* maps through case studies of a sample set of *AS*es. Marchetta et al. [94] propose a hybrid discovery tool, *MEasure the Router Level of the INternet (MERLIN)*, based on *mrinfo* and *traceroute* probes. The implementation employs a centralized server to increase the exploration coverage and to limit the probing redundancy, resulting in a more accurate router level map.

2.3.4 The Geographical Considerations in *BGP*

Oliveira et al. [95] propose *Geographically Informed inter-domain ROuting (GIRO)* that adds geographic location information into *BGP Update* messages to reduce the geographic distance between *BGP* speakers. Li et al. [96] inspect the relation between the *Geographic-Based Routing* (*GBR*) reducing geographical path length selected by *BGP* and the actual end-to-end data delivery performance. The simulation results shows that the *Gbps* did not show performance improvement in most cases although the practice can be theoretically improved by more than 50 percent. Similarly, Yihua He et al. [97] develop methods to identify missing *BGP* sessions to provide more accurate analysis, particularly of the *Peer-to-Peer* type. Their experiments show that 40 percent more edges and around 300 percent more *Peer-to-Peer* edges were identified, compared to commonly used data sets. In addition, Chen et al. [98] demonstrate that an approach to measuring the network that leverages

Peer-to-Peer systems can improve the understanding of AS topologies. They apply a comprehensive set of heuristics to the measurements derived from more than 992, 000 *IP*s and identified 23, 914 more links from the public view. Their results also indicate that (i) a substantial number of *Customer-to-Provider* links are not found by the public view (ii) and peering links can be missed by the tiers higher than the *Vantage Point*s in the *Internet* hierarchy.

Chapter 3

Construction of Secured *BGP* Networks

Despite efforts of researchers during past decades to mitigate or eliminate the *BGP* vulnerabilities causing the accidents in Section 2.2, the security appliance to *BGP* has not yet been stabilized or standardized because of the complexity in *BGP*. Intuitively, there is great interest in increasing the security of *BGP*. For example, the *United States* (*US*) government cites *BGP* security as a part of the national strategy to secure cyberspace [99]. The *Internet Engineering Task Force* (*IETF*) working group on *Secure Inter-Domain Routing* (*SIDR*) [100] has been investigating these security issues and defining practical solutions. In addition, *North American Network Operators Group* (*NANOG*) [101] also prominently discusses *BGP* security in their meetings. Nevertheless, the complexity and continuous demands in use of *BGP* make it evolved by numerous "*Add-on*s" instead of replacing the protocol to a new one. This patch-based development may cause the complex *Best Path Selection Algorithm* (*BPSA*); in other words, malicious routers have various choices to bring about *BGP* problems such as *IP Prefix Hijacking*. We believe that this type of vulnerability can be mitigated by supporting the *Validity* of byzantine robustness discussed in Section 2.3, and this topic is what we mostly focused on in this section.

3.1 The Past *BGP* Proposals on Security

Aiming at the *Validity* support, more than 15 *BGP*-securing solutions have been proposed by applying different technologies such as attestation, cryptography, database, penalty, or another protocol [102]. They are categorized into Non-*Public Key Infrastructure* (*PKI*)-based and *PKI*-based proposals in this section.

3.1.1 Non-*Public Key Infrastructure* (*PKI*)-based Proposals

Zhao et al. [103] found that a *Multiple Origin AS* (*MOAS*) conflict occurs when a particular prefix appears to originate from more than one *AS*. Intuitively, the valid reasons (e.g., *Multi-homing* or advertising routes to exchange points) and router misconfigurations can cause the *MOAS* conflicts. Most of the conflicts are short-lived, lasting only a few days. Wang et al. [104] propose a path-filtering approach to protect the routes to the critical top-level *DNS* servers. This is achieved by utilizing the high degree of redundancy in the top-level *DNS* system and the stability in network connectivity to the server locations. Xiaoliang Zhao et al. [105] enhance the *BGP* by adding a simple *MOAS* list to route announcements, in order to detect bogus route announcements from false origins. According to their words, the proposed solution is more robust in larger networks and can substantially reduce the impact of false routing announcements even with a partial deployment.

Lad et al. [106] develop the *Prefix Hijack Alert System (PHAS)*, which maintains prefix ownerships to evaluate the legitimacy of *BGP Update* messages. However, their contribution is disadvantaged in the cost and security of the *PHAS* server. To the face of misconfigurations and malicious attacks, Subramanian et al. [107] develop the *Listen and Whisper* method. Speaking

of which, the *Listen* detects unreachable prefixes with a low probability of false negatives, and the *Whisper* limits the percentage of nodes affected by a randomly placed isolated adversary to less than one percent. Whereas this proposal is advantaged in high deployability, supporting only detection (i.e., excluding correction) and their error-prone results weaken the contribution here.

Karlin et al. [108] develop *Pretty Good BGP* (*PGBGP*), which is an incrementally deployable modification to the *BGP* decision process. The *PGBGP* makes slow the propagation of *BGP* messages and analyzes the stability of an associated *BGP* network to construct a historical database, in which *PGBGP* compares database-entries to incoming *BGP* messages so as to discard unstable messages. Hu and Mao [56] improve the detection accuracy by combining analysis of passively collected *BGP Update* messages with data plane fingerprints of suspicious prefixes. In specific, data plane information is used in the form of edge network fingerprinting to disambiguate suspect *IP* hijacking incidences based on routing anomaly detection. Zhang et al. [109] propose *QBGP* that decouples the propagation and adoption of a path for data forwarding. The remarkable feature compared to the control plain-based anomaly detection is that *QBGP* does not use suspicious paths to forward data traffic but still propagates them in the routing system to facilitate attack detection.

3.1.2 *PKI*-based Proposals

Nordström and Dovrolis [110] identify several attack objectives and mechanisms provided that one or more *BGP* routers have been *compromised*. The researchers also review the existing and proposed countermeasures and argue that the *Secure BGP* (*S-BGP*) is heavyweight in computation load. The unrestricted nature of *BGP* provides routers significant flexibility in selecting routing paths and forwarding behavior. On the other hand, it possibly helps malicious routing attacks; for example, a *BGP* speaker can install a *null* route

for a set of prefixes and drop all the traffic destined to the networks. However, replacing *BGP* would be expensive and difficult. Although the replaceability can be argued by researchers, the design of a better inter-domain routing system should be considered. In other words, a fundamental trade-off does exist and should be examined between the *resilience* (i.e., trust-based network) and security of a routing protocol (i.e., verification-based network).

Ke Zhang et al. [111] concentrate on the *selective dropping attack* and show that the security solutions of *BGP* (e.g., *S-BGP* or *soBGP*) discussed in the paper are not sufficient to address the attack. In order to mitigate the "path exploration" phenomenon, Chandrashokar et al. [112] propose a mechanism called *forward edge sequence numbers* to annotate the *AS* paths with additional information providing path dependency. They also developed an enhanced path vector algorithm, *EPIC*, which prevents path exploration after failure and leads to faster convergence. However, it is not compared with *S-BGP* [113] and, to the best of our knowledge, we cannot find any remarkable advantage on this contribution compared to *S-BGP*. Zhao et al. [19] evaluate the impact of signatures, verifications, and certificates to the convergence time, message size, and storage. Their large-scale discrete-event simulation with *Secure BGP* (*S-BGP*), *Signature Amortization (S-A)*, and *Sequential Aggregation Signatures (SAS)* reveals that more efficient cryptographic operations will be available to improve the performance in terms of convergence time, message size, or storage cost.

Goldberg et al. [114] prove that finding the most damaging strategy is *Non-deterministic Polynomial-time Hardness* (*NP-hard*), and show how counterintuitive strategies (e.g., announcing longer paths, announcing to fewer neighbors, or triggering *BGP* loop-detection) can be used to attract even more traffic. While secure routing protocols can blunt traffic attraction attacks, they found that export policies are very effective attack vector that those protocols do not cover. They suggest that secure routing protocols (e.g., *S-BGP*, *soBGP*) should be deployed in combination with mechanisms that

police export policies (e.g., defensive filtering). However, policing export policies is a significant challenge in practice as evidenced by accidents [48]. Boldyreva and Lychev [115] design a security definition for routing path vector protocols by studying, generalizing, and formalizing numerous known threats. Regarding the definition, *S-BGP* satisfies two out of the three goals, and *soBGP* failed to meet two security goals. One of their main criticisms of *S-BGP* is that it is very inefficient in terms of processing and communication overhead. There have been various proposals for more efficient ways than the route attestation mechanisms in *S-BGP*.

Due to weaknesses such as high computational cost or potential security vulnerability, Xiang et al. [116] propose *Fast Secure BGP (FS-BGP)*, which can secure *AS*-paths and prevent *Prefix Hijacking* by signing critical *AS*-path segments. They demonstrate that *FS-BGP* achieves a similar level of security as *S-BGP*, but with much higher efficiency. Their experiments show that the cost of signing and verification in *FS-BGP* can be reduced by orders of magnitude, compared to *S-BGP*. Oorschot et al. [117] trade off the strong security guarantees of *Secure BGP* (*S-BGP*) for presumed-simpler operation. They accordingly develop a decentralized trust model for verifying the property of *IP* prefix origin, which utilizes a single-level *Public Key Infrastructure* (*PKI*) for *ASN* authentication, as well as a rating-based stepwise approach for the *Path Attribute* verification. Bellovin et al. [118] argue that it is problematic to partially deploy a securing solution to the *BGP* due to the existence of bypasses. Zhu et al. [119] propose the notion of *AS* Alliance, which is a natural community structure self-organized based on commercial, political, or other social relationships, and taking advantage of the power-law and rich-club features of *AS*-level topology. A new trust model, *Trust Translator Model (TTM)*, is designed based on *AS* Alliance to improve the security of *BGP*. This model yields much less memory overhead and a shorter validation chain than the traditional solutions. Correspondingly, they develop a novel SE-*BGP* (Security Enhanced *BGP*) protocol based on the *TTM*. Lychev et al.

[120] found that *BGP* solutions providing *Path Validation* introduce only meager benefits over *Origin Validation* when those solutions facilitate the popular policies that are derived in their survey of 100 network operators. They also show how the partial deployment of the solutions can produce new vulnerabilities into the routing system.

3.1.3 Deduction to the *Resource Public Key Infrastructure* (*RPKI*) from Past *BGP* Security Proposals

BGP speakers facilitate *BGP Update* messages to draw its network diagram (i.e., to determine best routes to destinations). Namely, if the integrity of corresponding information in *BGP Update* messages is verified, then the *Validity* in byzantine robustness can be provided in *BGP* networks. However, misconfigured *BGP Update* messages can be consciously or unconsciously generated by legitimate origin *AS*es, as discussed in Section 2.1. It is noted that a single *ISP* misconfiguration error may have the disproportionate impact of the global *Internet* routing [121], and that configuration errors are inevitable in *BGP*. Mahajan et al. [30] found that 200 to 1200 *IP Prefix*es, which occupied 0.2 to one percent of a regular *BGP* routing table at that time, suffer from misconfiguration each day. When considering the growth rate of active *BGP* entries in a routing table [122], approximately 1000 to 6000 prefixes are misconfigured and suffer at this moment. That is to say, it is actually required to "validate" the *NLRI* and *AS_Path* attribute in *BGP Update* messages, of which the format is provided in Figure 1-4. These functionalities are referred to as the *Origin Validation* and *Path Validation*. The ***Origin Validation*** is to validate all *NLRI* fields in a *BGP Update* message to examine if the origin *AS* of the message is authorized to announce the given *Network Layer Reachability Information* (*NLRI*). The ***Path Validation*** is to verify the *Path Attribute* in a *BGP Update* message to check if the authenticity of attribute is

ensured.

Security and correctness cannot be guaranteed with the non-*PKI*-based security methods discussed in Section 3.1.1, of which misjudgments may disturb the *BGP* routing system. The *Public Key Infrastructure* (*PKI*) has been adopted by the *Internet Engineering Task Force* (*IETF*), and *Regional Internet Registries* such as *African Network Information Center* (*AfriNIC*), *Asia Pacific Network Information Centre* (*APNIC*), *American Registry for Internet Numbers* (*ARIN*), *Latin America and Caribbean Network Information Centre* (*LACNIC*), and *Réseaux IP Européens Network Coordination Centre* (*RIPE NCC*), have started offering related services. It is concluded that the *PKI* is an essential part of the *Internet* security framework. All and most solutions respectively discussed in Section 3.1.1 and Section 3.1.2 provide only a few facets of the byzantine robustness due to the complexity of *BGP* [102]. The *Secure BGP* (*S-BGP*) [123] and *Inter-domain Route Validation* (*IRV*) are only solutions that cryptographically and fully protects *BGP* networks [20, 102]. The *S-BGP* provides more comprehensive security than *IRV* by motivating the hierarchical *Internet* structure (i.e., the *IP Prefix* allocation) for its *PKI*. The *S-BGP* protocol [113] is the most representative one of the past *PKI*-based proposals, and its associated architecture had been standardized in 2003 by the *IETF*. The main problems of *S-BGP* are the cost of producing, storing and distributing attestations, and the need to bootstrap *PKI* [124] in such a way that it is not possible to be deployed in the real-world [19, 113]. Speaking of the deployability, it may be discussed that the references are old and current hardware possibly overcomes the overhead. However, we argue that the improvement in hardware resource is "only" positive facet; for example, the increment of traffic and *RIB* entries [125] in *BGP* should be appropriately considered to examine the feasibility in terms of hardware resources. More approaches are then proposed on the basis of *S-BGP* such as *Secure Origin BGP* (*soBGP*) [126] or *Pretty Secure BGP* (*psBGP*) [117]. These solutions differentiate the structure of *S-BGP* in its trust model or

in the trade-off between its security coverage and performance in order to mitigate the *S-BGP* overhead. In specific, *S-BGP* is based on a hierarchical trust model, *soBGP* leverages web-of-trust model, whereas *psBGP* introduces a decentralized model introducing additional neighboring relationships. The trust models in *soBGP* and *psBGP*, however, do not take advantage of the prominent inherent topology of the *Internet* in *AS*-level interconnection [119]. Moreover, Zhao et al. [19] found from their large-scale network simulation and analysis that the more they look into the performance of these proposals, the more issues and potential ways to improve performance are observed.

Among numerous solutions that have been proposed to protect the *NLRI* (i.e., one or more origins) and *Path Attribute* (i.e., an *AS* path) in a *BGP Update* message, remarkable proposals in validating origins and paths are the *Resource Public Key Infrastructure* (*RPKI*) and *BGPsec* respectively. The ***RPKI*** [127], which is stemmed from *S-BGP*, examines if an origin *AS* in the *Path Attribute* really owns the prefixes defined in the *NLRI* field by comparing them with the *Route Origin Authorization*s (*ROA*s) [128, 129]. The ***BGPsec*** accepts the *RPKI* in validating origins and validates paths by scrutinizing the signatures in *BGPsec Path Attribute*s [2], in which every *BGP* session will be signed and checked if they originated from and received by a legitimate *AS*. Detailed discussions regarding *BGPsec* will be provided in Chapter 4. The *RPKI*, which is also known as *Resource Certification*, is specialized to secure *BGP* by exchanging X.509-based resource certificates [130] to verify origins in *BGP* networks based on a threat analysis [28]; note that the pure *RPKI* does not support *Path Validation*. At this moment, *RPKI*-capable software routers are deployed in the discrete sets of small areas all over the world; in particular, around five percent of all unique *IPv4* prefix/origin pairs is currently protected by leveraging *RPKI* according to the monitor [131]. The *RIPE NCC* and *LACNIC* are encouraging the other three *RIR*s to leverage *RPKI* by enthusiastically publishing *RPKI*-capable *BGP* routers under their administrations.

3.2 The Implementation of an *RPKI*-capable *BGP* Software Router

Due to the deployability and popularity of *RPKI* discussed in Section 3.1.3, we have chosen it to fully secure the vulnerable *BGP* network as *S-BGP* does. For that purpose, a *BGP* router supporting *RPKI* was required to construct and analyze *RPKI*-capable *BGP* networks. In addition, we decided to utilize a *Software Router* rather than a hardware router, due to the cost-effectiveness and flexibility of software. Among the previously developed software-based routing solutions that are *Quagga* [132], *BIRD* [133], *XORP* [134], and *Vyatta* [135], *Quagga* was chosen because only *Quagga* had supported *RPKI*, and the *RPKI*-capable *Quagga* was called Quagga*SRx* that is implemented by *National Institute of Standards and Technology* (*NIST*) [136].

3.2.1 The Design of *BIRD-SRx*

Filip et al. [137] show in performance comparison between *Quagga* and *BIRD* that *BIRD* occupies a few times fewer *CPU* clocks and memory than *Quagga*. So as to take the advantage of hardware resources and to provide the *RPKI* capability, we here proposed a migration of the *RPKI* implementation of *Quagga* into *BIRD*. It is documented that the *RPKI* functionality is implemented as an independent module named *SRx* in Quagga*SRx* [136]. Figure 3-1 describes this migration idea. Basically, a Quagga*SRx*-based border (i.e., *BGP*) router is able to communicate with an *SRx* server that is capable of validating origin *AS*es in *IP Prefix* ownership based on the *ROA*s provided by one or more virtual *validated cache*s. The core technology to communicate with an *SRx* server is an *SRx* proxy on the perspective of *Quagga*. In other words, if the *SRx* proxy can be embedded into *BIRD*, then this implementation will offer much better performance than Quagga*SRx* in the

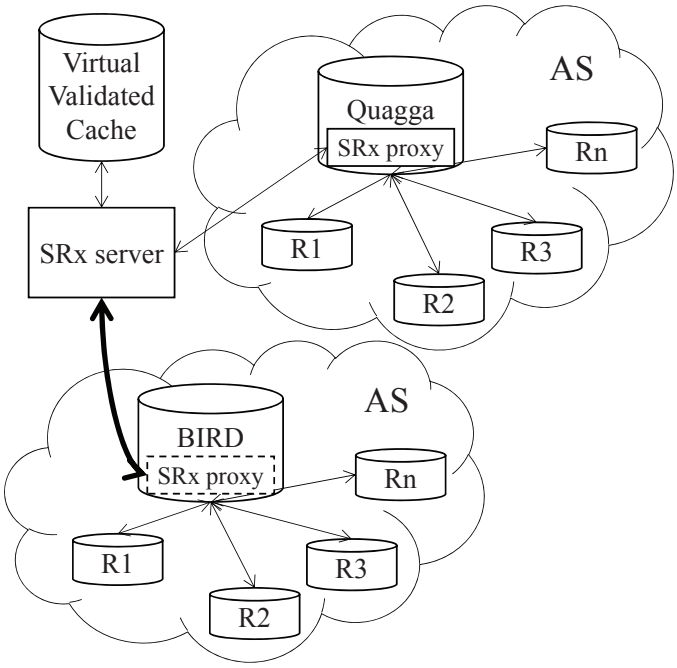


Figure 3-1: The Migration Idea of *SRx* into a *BIRD* Software Routing Daemon

same computing environment because of the difference in performance between them. (i) Nonetheless, we found from our inspection of Quagga*SRx* that *SRx* highly depends on *Quagga* in functions as opposed to the description of *NIST*. In other words, *BIRD* requires following the functional flow of Quagga*SRx* by facilitating all *Quagga*-initiated functions in order to have the proxy integrated. Note that the version of Quagga*SRx* used here is 0.99.16 with the *SRx* server 0.2.0. (ii) We also found from Quagga*SRx* that *SRx* is unable to interact with an actual cache server in such a way that we cannot directly exploit the implementation in a practical simulation. To be specific, *SRx* can be coupled with only virtual *validated cache* in which *ROA*s [128] should be artificially injected rather than using real-world *ROA*s extracted from an actual *validated cache*. As a result, Quagga*SRx* could not be leveraged to build a "practically" *RPKI*-capable *BGP* network although its virtual environment contributes to the flexibility and scalability of simulation environments. (iii) Moreover, virtual cache servers, virtual *validated cache*, as well as their artificial *ROA*s in Quagga*SRx*-based networks cannot be shared between Quagga*SRx* routers. In

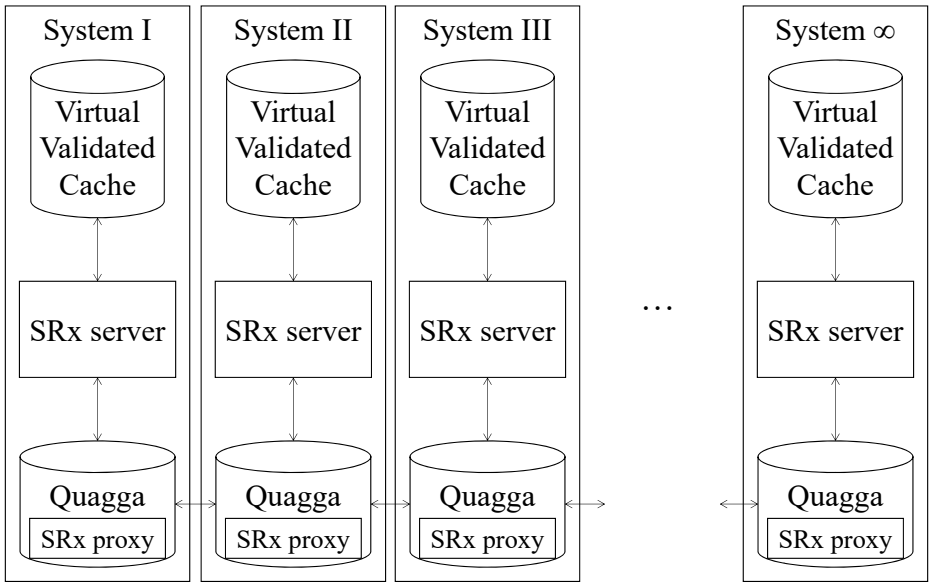


Figure 3-2: The Structural Overview of the Quagga*SRx*-based *BGP* Network

other words, each Quagga*SRx* should have its own virtual *validated cache* and *SRx* server, and the *ROA*s in each virtual *validated cache* can be different from that in the other *validated cache*; that is, the same *NLRI* can be sometimes considered *valid* or *invalid* depending on a *validated cache* that is used for *Origin Validation*. In addition, it cannot be guaranteed if *ROA*s in use are latest ones due to the manual injection of them. These behaviors violate the *IETF* standard [127] and are visualized in Figure 3-2.

3.2.2 The Design, Implementation, and Simulation of *RTR-BIRD*

Alternatively to have an *RPKI*-capable *BGP* software router, we have determined to implement a module to support *RPKI* in the router. The implementation is expected (i) to necessitate fewer hardware resources, (ii) to leverage real-world *ROA*s for *Origin Validation*, (iii) and to follow the *IETF* standard [127]. For this development, the *SRx* module of Quagga*SRx* is substituted to the *RTRlib*. The *RTRlib* [138] is an *open-source C* library that implements the *RPKI*/RTR protocol. Wählisch et al. [139] observe that

most invalid prefixes are most likely the result of misconfiguration and that a relatively small overhead of *Origin Validation* does occur on commodity hardware; in specific, five percent more *Random Access Memory* (*RAM*) than required for full *BGP* table support. Our implementation, which is used to replace the *SRx* proxy of the design discussed in Section 3.2.1, is referred as to *RTRPKI*. The *RTRPKI* module and its communication to a regular *BIRD* software router are the core contribution in this part of the study.

Figure 3-3 visualizes the functional architecture of *RTR-BIRD* in a high-level view. The *RTR-BIRD* and *RTRPKI* are generally located in the same system and share the *Inter-Process Communication* (*IPC*) memory (i.e., the shared memory in this case) to exchange *NLRI*. In this version of implementation, *RTRPKI* is operated only when it is invoked by *RTR-BIRD*. In other words, *RTRPKI* will first synchronize itself with an associated *RTRlib* cache server in its life cycle. Once the synchronization is completed, the *RTRPKI* is able to retrieve real-world *ROA*s from one or more practical *validated cache*s. The *RTRPKI* utilizes those *ROA*s to examine *NLRI* in the shared memory; that is, the *Origin Validation*. Every single validation result is stored in the shared memory and is documented in a locally existing file, which consists of an *ASN*, *IP Prefix*, validation result (i.e., *valid*, *invalid*, *not-found*), and timestamp. After these processes, the *RTRPKI* is self-terminated and waits for the next call.

Once a response is provided by *RTRPKI* in the shared memory, *RTR-BIRD* immediately consumes it to determine whether to update its *RIB* with a *BGP Update* message that is subjected to the *Origin Validation*. The *RTR-BIRD* is developed based on the version 1.3.11 of *BIRD* software routing daemon [133], and *RTRPKI* is founded upon the version 0.2.3 of *RTRlib* [138]. Figure 3-4 visualizes our implementation that resolves all three problems discussed in Section 3.2.1. In specific, the *SRx* proxy in Quagga*SRx* is replaced with *RTRPKI* in *RTR-BIRD*. The virtual validate cache implemented by *NIST* is substituted to the *RIPE NCC RPKI Validator* that acts as a cache server

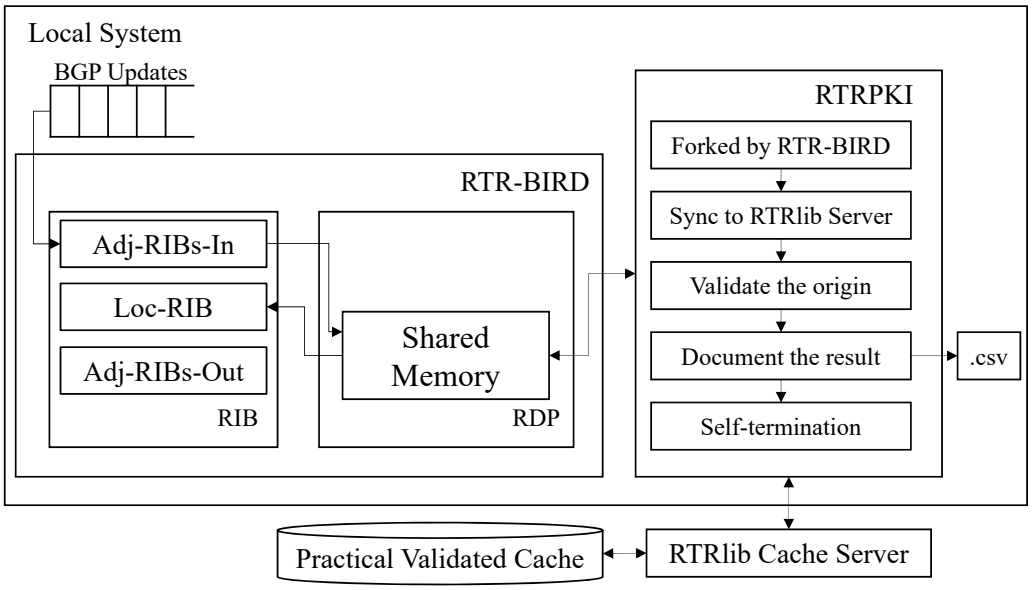


Figure 3-3: The Systematic Architecture of *RTR-BIRD* (i.e., *RTR-BIRD* v1.0)

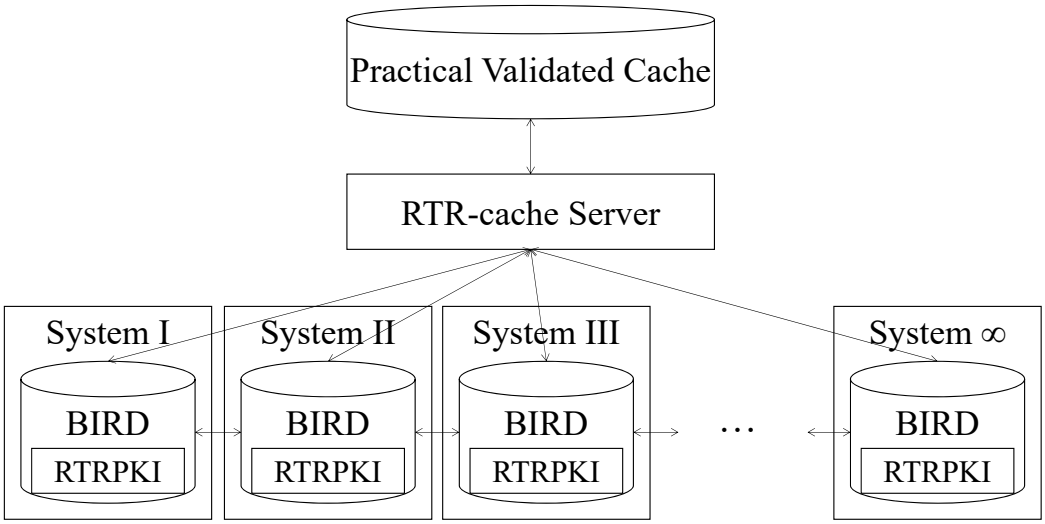


Figure 3-4: The Structural Overview of the *RTR-BIRD*-based *BGP* Network

and provides connections to real-world *validated caches* through the *Trust Anchor Locators* (*TALs*). This cache server also supports the *IETF* standard [127] so our implementation conforms to the standard as well. It means that *ROAs* of *validated caches* can be shared between *BGP* speakers and can be automatically updated to the latest ones. Note that the version of *RIPE NCC RPKI Validator* used here is 2.0.

We exported 6055 *IPv4*-based real-world *ROAs* via the *RIPE NCC RPKI*

Validator and re-formatted all of the *ROA*s into *BGP Update* messages; in specific, one *NLRI* is included per update message. We then sent out those update messages to the *RTR-BIRD*. As discussed in this section, *RTRPKI* had validated all 6055 messages and logged all validation results. Note that when those update messages are validated by *RTR-BIRD* all of them must be considered *valid* in *Origin Validation* because they are extracted from *valid ROA*s. Figure 3-5 shows the simulation results in accuracy (i.e., Figure 3-5a) and time (i.e., Figure 3-5b). In the simulations, we made and examined few more variations of *RTR-BIRD*, which are *Origin Validation Program* (*OVP*), *OVP* with partial *Origin Validation Cycle* (*OVC*), and *OVP* without synchronization. The *OVP* is basically an *RTRPKI* but cannot be associated with a *BGP* router (i.e., *BIRD*); in other words, it is simply able to verify *NLRI*. That is to say, this implementation ignores the forking step in *RTRPKI* depicted in Figure 3-3. The *OVP* with partial *OVC* assumes that the synchronization to an *RTRlib* cache server has been finished. In other words, this version neglects the forking and syncing steps in the *Origin Validation Cycle* (*OVC*). The last version, *OVP* without synchronization, invokes the synchronization but it does not wait for the completion of it; intuitively, the sync can be completed in the middle of validations in time with high probability. Note that the three different types of *OVP*s take a text-based file instead of *BGP Update* messages due to the lack of routing capability; in other words, these developments should be slightly faster than *RTRPKI* because of the forking process in *RTR-BIRD*.

3.2.3 The Analysis of *RTR-BIRD*

For a deeper analysis of Figure 3-5, Table 3.1 demonstrates the number of valid entries and delay for *Origin Validation*s per 1000 *BGP Update* messages until the total number of updates (i.e., 6055 updates) in the simulation. Both *RTR-BIRD* and *OVP* shows more than 98 percent accuracy in "practical" origin validation all along in Figure 3-5a. The less than two percent errors

Total *BGP* Updates	**Aggregated *NLRI*s Considered Valid**			
	RTR-BIRD	*OVP*	*OVP* without *SC*	*OVP* with partial *OVC*
1000	986	986	0	986
2000	1978	1978	0	1978
3000	2969	2969	947	2969
4000	3956	3957	1935	3957
5000	4937	4939	2917	4939
6055	5977	5976	3947	5976

(a) The Quantification of Simulation Results in Figure 3-5a

Total *BGP* Updates	**Aggregated Time Measured in Seconds**			
	RTR-BIRD	*OVP*	*OVP* without *SC*	*OVP* with partial *OVC*
1000	1982	1778	0.323	0.369
2000	3972	3691	0.705	0.794
3000	5904	5495	1.138	1.191
4000	7873	7351	1.500	1.576
5000	9847	9243	1.889	1.950
6055	11, 869	11, 195	2.322	2.388

(b) The Quantification of Experimental Results in Figure 3-5b

Table 3.1: The Quantified Results of Simulations Demonstrated in Figure 3-5

might originate from the change of *ROA*s in real-world *validated cache*s within the timeframe more than three hours. Another noticeable point in Figure 3-5a is the accuracy of *OVP* without synchronization. It introduces zero percent in accuracy for the first 2067 *NLRI*s, and we found that during the time when they were being validated the synchronization to a cache server has not been completed, resulting in the following observation.

Observation 3.2.1. On the perspective of *RTRPKI*, the synchronization to an associated cache server should be accomplished before performing *Origin Validation*s.

In Figure 3-5b, the *RTR-BIRD* and *OVP* took around 3.3 and 3.1 hours to validate 6055 *NLRI*s. The difference between those two types is only the forking process; in other words, most delays originated from the *Origin Validation*s themselves, and the communication between *RTR-BIRD* and *RTRPKI* produces relatively negligible delay. However, the difference between *RTR-BIRD* and *OVP* in delay that is 674 seconds to validate 6055 origins can be significant, provided that *BIRD* generally processes a *BGP Update* message

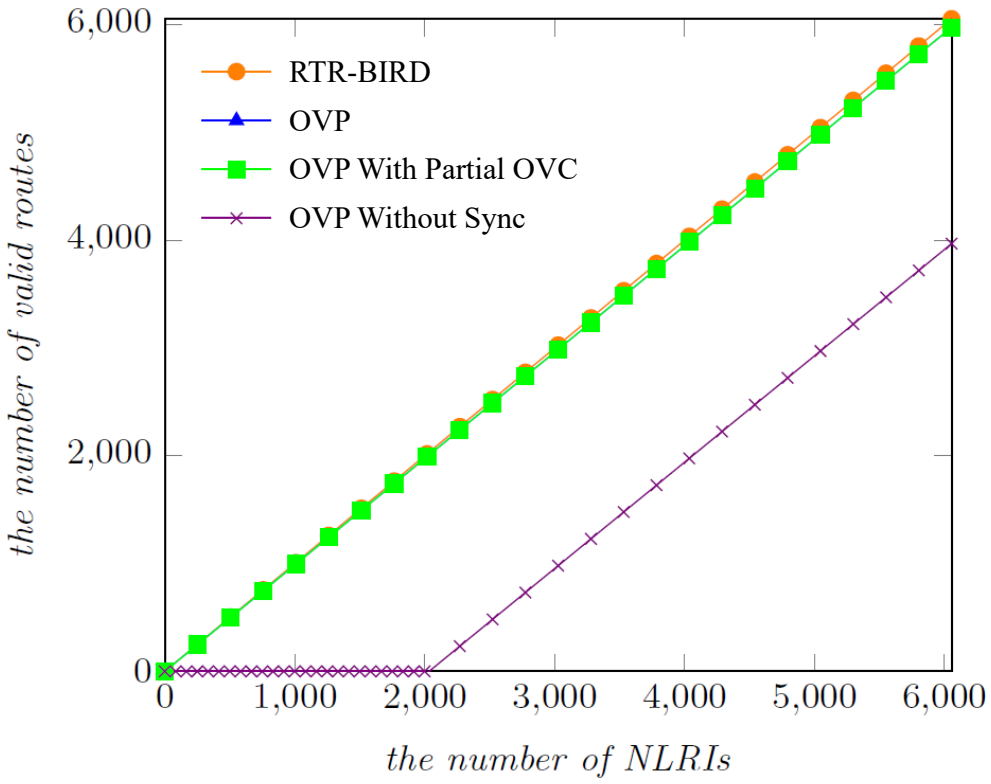
in about 0.4 milliseconds. In other words, approximately 111 milliseconds is additionally required by *RTR-BIRD* to process a single *BGP Update* message. We found that this delay is caused by sharing the *IPC* memory for inbound and outbound traffic in *RTR-BIRD*, leading us to the following observation.

Observation 3.2.2. In order to reduce the delay originating from *IPC*, a discrete *IPC* memory should be provided for each unidirectional traffic.

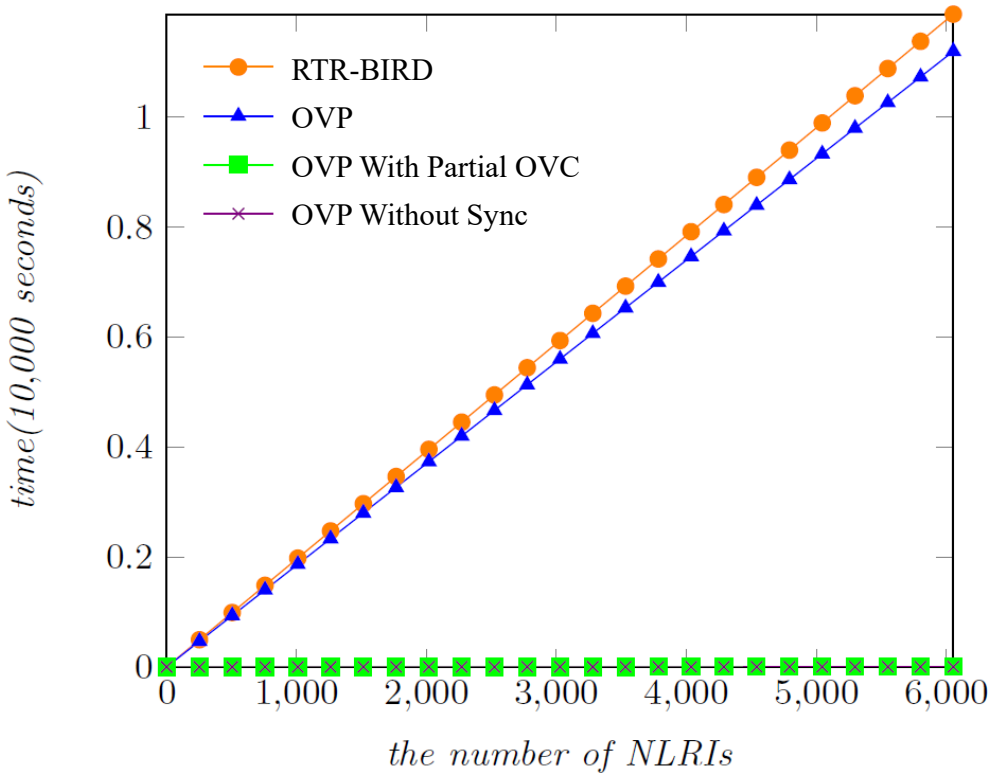
In addition, regardless of the validation accuracy, *OVP* with partial *OVC* and *OVP* without synchronization validated all 6055 *NLRI*s in 2.322 and 2.388 seconds. These two implementations are distinguished from the other two types in disregarding synchronization so we could reach the following observation.

Observation 3.2.3. The most significant factor in *RTRPKI* is the synchronization delay.

Figure 3-6 visualizes the updated architecture of *RTR-BIRD* in consideration of Observation 3.2.1, 3.2.2, and 3.2.3. Speaking of which, *RTRPKI* is decoupled with *RTR-BIRD* and is individually operated to separate the synchronization process from the *OVC*. In other words, an *RTRPKI* process should be manually invoked and of which the synchronization should be achieved before *RTR-BIRD* runs. As another remarkable modification, *RTR-BIRD* facilitates two message queues for each unidirectional traffic instead of the shared memory in Figure 3-3.



(a) The Count of *BGP Update* Messages Considered *Valid*



(b) The Measured Time to Process *BGP Update* Messages in 10, 000 Seconds

Figure 3-5: The Simulation Results of *RTR-BIRD* to Process 6055 *BGP Update* Messages that are Generated Based on *valid ROA*s

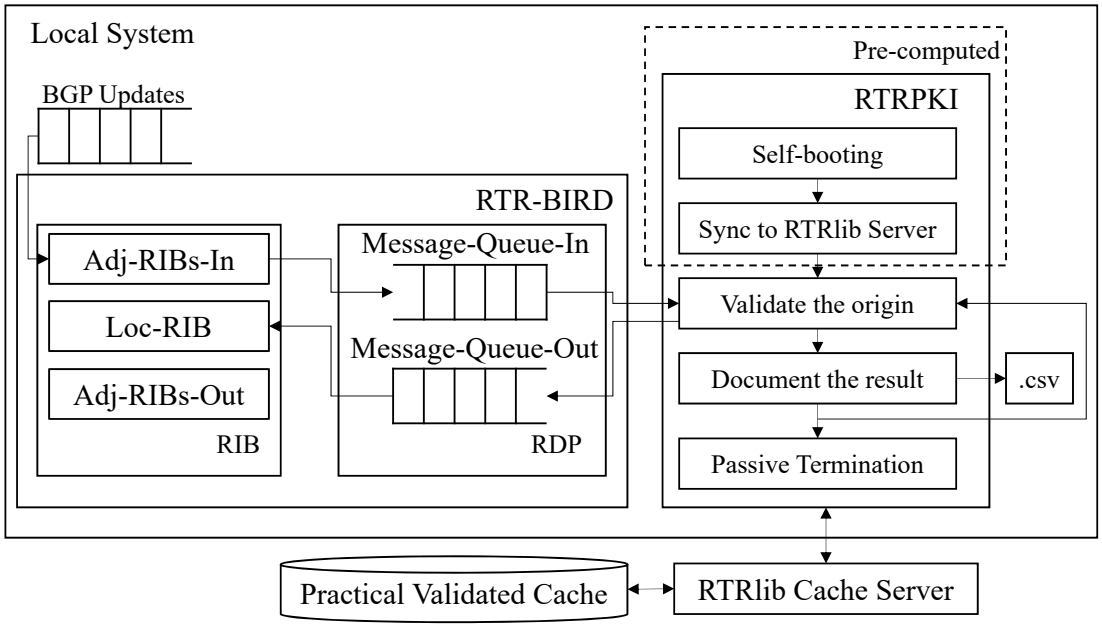


Figure 3-6: The Systematic Architecture of *RTR-BIRD* (i.e., *RTR-BIRD* v1.1) that is Enhanced in Performance and Accuracy

We simulated the updated version of *RTR-BIRD* with the *NLRI*s facilitated in Section 3.2.2. Note that the original *RTR-BIRD* is referred to as *RTR-BIRD* v1.0, and the newer version is referred to as *RTR-BIRD* v1.1. Experimental results of *RTR-BIRD* v1.1 are visualized in Figure 3-7 and are integrated with the simulations performed in Section 3.2.2, respectively in accuracy and performance. The *RTR-BIRD* v1.1 as the first improvement yields 100 percent accuracy in its inspection as shown in Figure 3-7a. The other three types that validated *NLRI*s after an associated cache server is entirely synchronized produce 98.4 to 99.2 percent of accuracy in *Origin Validation*. The remaining type (i.e., *OVP* without *Synchronization Completion* (*SC*)) is ended with around 65 percent accuracy in validating origins; seemingly, around 35 percent of difference originated from the fist 2000 validations that were performed without *ROA*s. Figure 3-7b represents the duration of every single *Origin Validation* attempt derived in each *RTRPKI* type; in the figure, the fluctuations of delay in microseconds are visible. The boldest lines of *OVC* without *SC*, *OVP* with partial *OVC*, and *RTR-BIRD* v1.1, are formed

Total *BGP* Updates	**Aggregated *NLRI*s Considered Valid**				
	RTR-BIRD v1.0	*OVP*	*OVP* without *SC*	*OVP* with partial *OVC*	*RTR-BIRD* v1.1
1000	986	986	0	986	1000
2000	1978	1978	0	1978	2000
3000	2969	2969	947	2969	3000
4000	3956	3957	1935	3957	4000
5000	4937	4939	2917	4939	5000
6055	5977	5976	3947	5976	6055

(a) The Simulation Results of *RTR-BIRD* v1.1 Integrated with Table 3.1a in Accuracy

Total *BGP* Updates	**Aggregated Time Measured in Seconds**				
	RTR-BIRD v1.0	*OVP*	*OVP* without *SC*	*OVP* with partial *OVC*	*RTR-BIRD* v1.1
1000	1982	1778	0.323	0.369	0.341
2000	3972	3691	0.705	0.794	0.995
3000	5904	5495	1.138	1.191	1.906
4000	7873	7351	1.500	1.576	2.662
5000	9847	9243	1.889	1.950	3.665
6055	11, 869	11, 195	2.322	2.388	4.453

(b) The Simulation Results of *RTR-BIRD* v1.1 Integrated with Table 3.1b in Measurement

Total *BGP* Updates	**Aggregated NLRIs Considered Valid**		**Aggregated Time Measured in Seconds**	
	RTR-BIRD v1.0	*RTR-BIRD* v1.1	*RTR-BIRD* v1.0	*RTR-BIRD* v1.1
1000	986	1000	1982	0.341
2000	1978	2000	3972	0.995
3000	2969	3000	5904	1.906
4000	3956	4000	7873	2.662
5000	4937	5000	9847	3.665
6055	5977	6055	11, 869	4.453

(c) The Quantitative Comparison between *RTR-BIRD* v1.0 and *RTR-BIRD* v1.1

Table 3.2: The Quantified Results of Simulations Demonstrated in Figure 3-7 (i.e., including the results of *RTR-BIRD* v1.1)

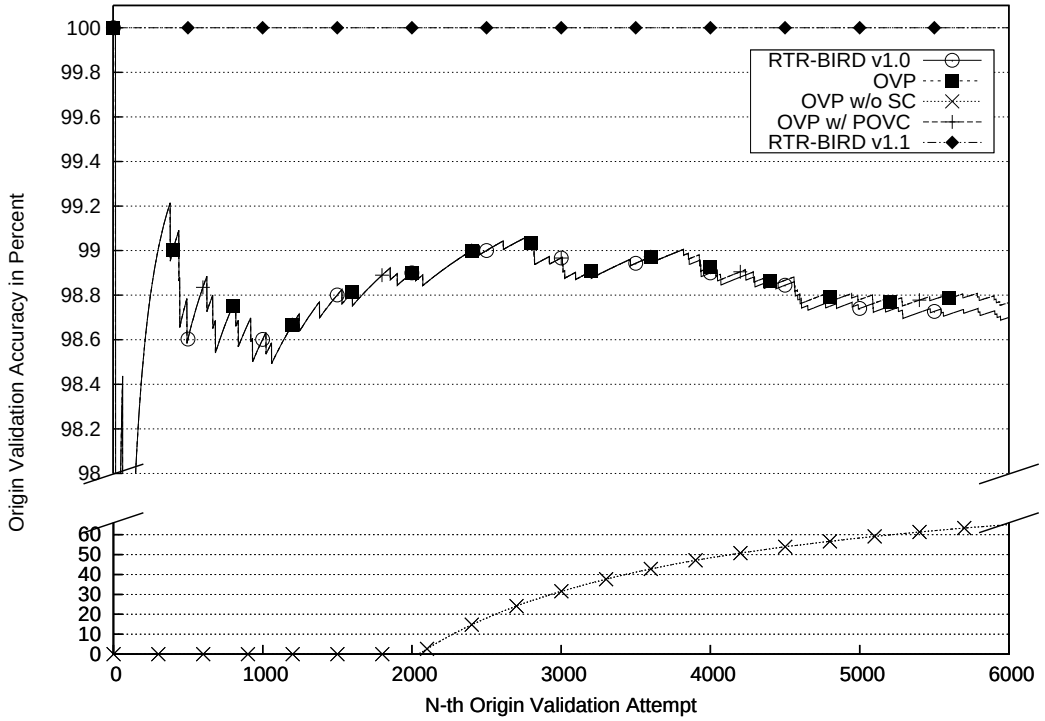
around 30 microseconds and located at the bottom of the graph for both types, proving that Observation 3.2.3 is true. In addition, the only difference between *OVP* with partial *OVC* and *RTR-BIRD* v1.1 is the existence of *IPC* with *RTR-BIRD*, as explicitly same as the difference between *OVP* and *RTR-BIRD* v1.0 discussed in Section 3.2.3 (i.e., for Observation 3.2.2). In Table 3.2b, the difference between them is 2.065 seconds produced while 6055 *NLRI*s are validated; that is the *IPC* delay is reduced by 99.7 percent (i.e., 674 to 2.065 seconds) so Observation 3.2.2 is proven.

3.3 The Discussions for Comparison between *RTR-BIRD* v1.0, *RTR-BIRD* v1.1, and Quagga*SRx* in Performance

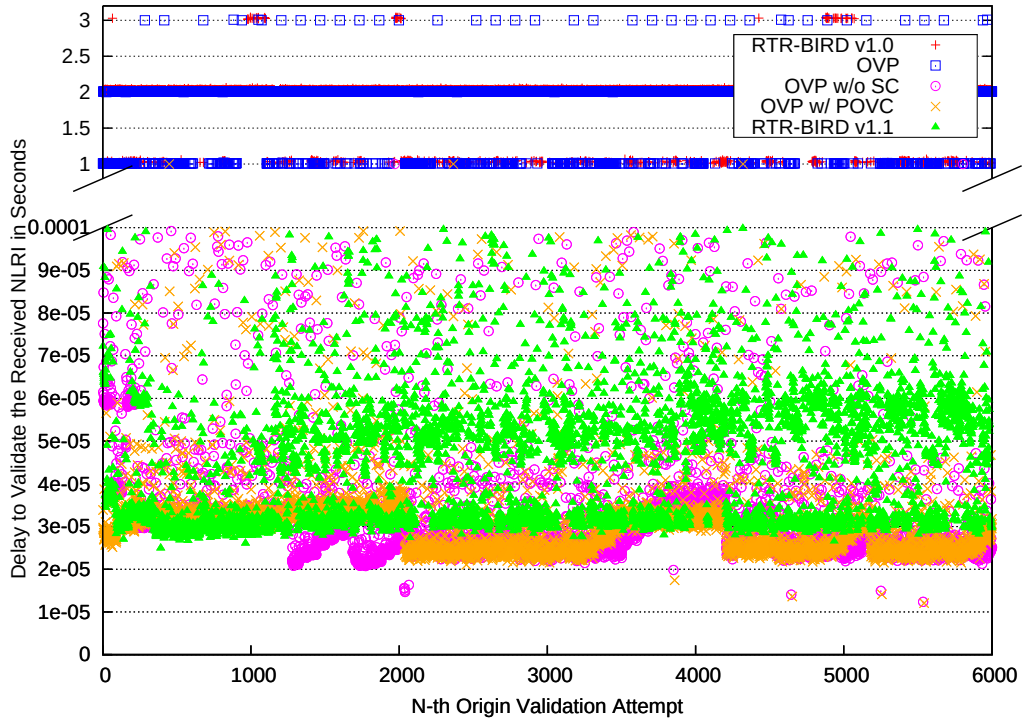
Table 3.2c shows that *RTR-BIRD* v1.1 is approximately 2665 times faster than *RTR-BIRD* v1.0 in *Origin Validation* and provides perfect accuracy by eliminating around two percent error shown in *RTR-BIRD* v1.0. It also reveals that *RTR-BIRD* v1.1 processes a *BGP Update* message in 0.7 milliseconds; by the way, a regular *BIRD* software router takes around 0.4 milliseconds to process a *BGP Update* message. However, we noticed that the simulations were performed without technically optimizing the code of *RTR-BIRD* v1.1; that is the *RTR-BIRD* v1.1 included a bunch of debug codes at that time. Accordingly, the *RTR-BIRD* v1.1 has been re-factored and then compared to *RTR-BIRD* v1.0 as well as to Quagga*SRx*; the previously created 6055 *BGP Update* messages are exploited again in this comparison. Quagga*SRx* used here is the version 0.3.1.1 that is coupled with the *SRx* 0.3.0.3. The 6055 *BGP Update* messages were propagated to each *RPKI*-capable *BGP* software router—*RTR-BIRD* v1.0, *RTR-BIRD* v1.1, and Quagga*SRx* v0.3.1.1—which intuitively have validated 6055 *BGP Update* messages in the order of receipt and documented results to a file as performed in the previous simulations. Note that, for the simulation of Quagga*SRx*, we manually put the 6055 *ROA*s into its virtual *validated cache*, and the timestamps are inserted at the beginning and ending point of the *BGP Update*-handling procedure in Quagga*SRx*. Figure 3-8 visualizes the validation results of *RTR-BIRD* v1.1 compared to Quagga*SRx* and *RTR-BIRD* v1.0. The *RTR-BIRD* v1.1 has validated most *BGP Update* messages in origins around 30 microseconds in Figure 3-8a. The other validations of updates have been performed in the range of 35 to 90 microseconds, resulting in 384 microseconds in average to validate an origin

(i.e., a single *NLRI*); that is, the re-factorized version of *RTR-BIRD* v1.1 produces no delay to process a *BGP Update* message compared to a regular *BIRD* router. On the other hand, Quagga*SRx* has validated most *BGP Update* messages in 20 microseconds per *BGP Update* message in the same figure. Most of the other updates have been validated in the range of 70 to 200 microseconds per *NLRI*, resulting in 51 microseconds in an average of *Origin Validation*s.

According to the simulation results given in Figure 3-8a, the Quagga*SRx* seems to be almost eight times faster than *RTR-BIRD* v1.1 in average. However, it does not mean that the Quagga*SRx* is actually faster than *RTR-BIRD* v1.1 in performance because of the difference in simulation environment between them as depicted in Figure 3-9. Speaking of which, the fully virtualized network environment of Quagga*SRx* does not produce any delay because all network components reside in a local computer system, whereas the *RTR-BIRD* v1.1-based network shows network delay since each network component is located exclusively in a different place. Although it was possible to remove the delay between the server and *RTR-BIRD* v1.1-based network in Figure 3-9b by putting them in the same system, the delay to access one or more real-world *validated cache*s could not be reduced or eliminated. In other words, the existence of delay in *RTR-BIRD* v1.1 negatively biases the performance of our implementation. In addition, we found from our simulation that Quagga*SRx* v0.3.1.1 significantly declines its performance for the first 650 *Origin Validation*s. Incidentally, this performance degradation also occurs on a regular basis and short period of time, resulting in clear vertical lines in Figure 3-8a. These remarks will be critical to those who are interested in improving the Quagga*SRx* in performance. Moreover, the comparison between *RTR-BIRD* v1.0 and *RTR-BIRD* v1.1 is much more obvious as shown in Figure 3-8b. The *RTR-BIRD* v1.1 processed a *BGP Update* message in 0.08 millisecond at worst whereas *RTR-BIRD* v1.0 took 1.1 seconds at best to process a *BGP Update* message.

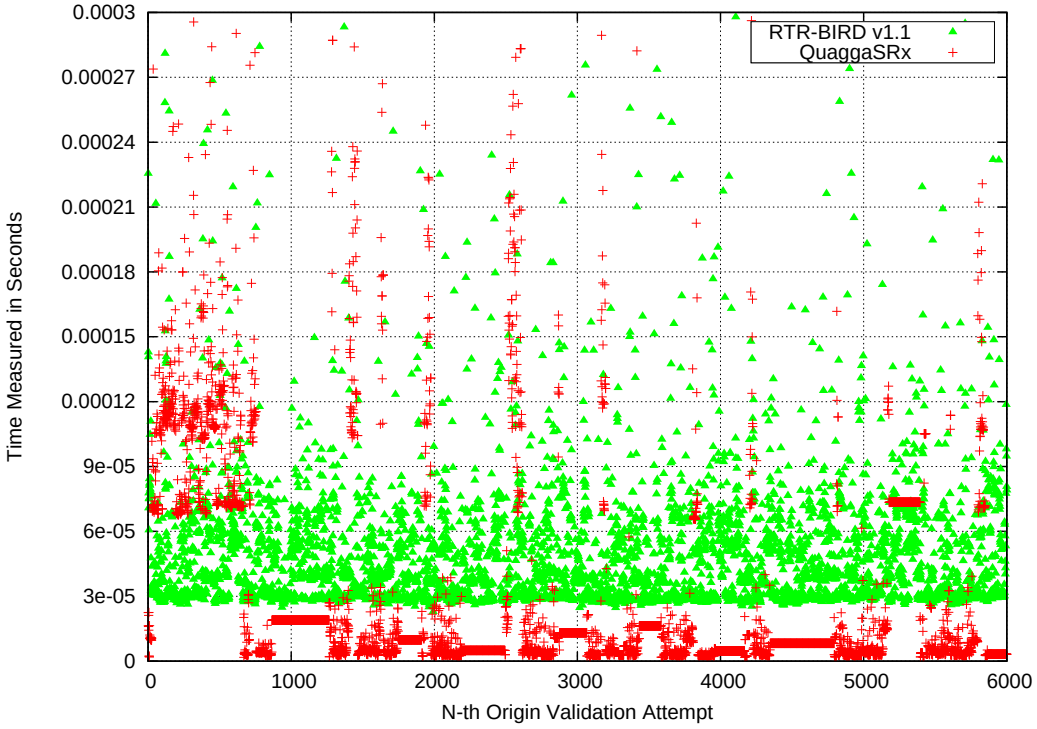


(a) Comparison in Performance between *RTRPKI* Variations

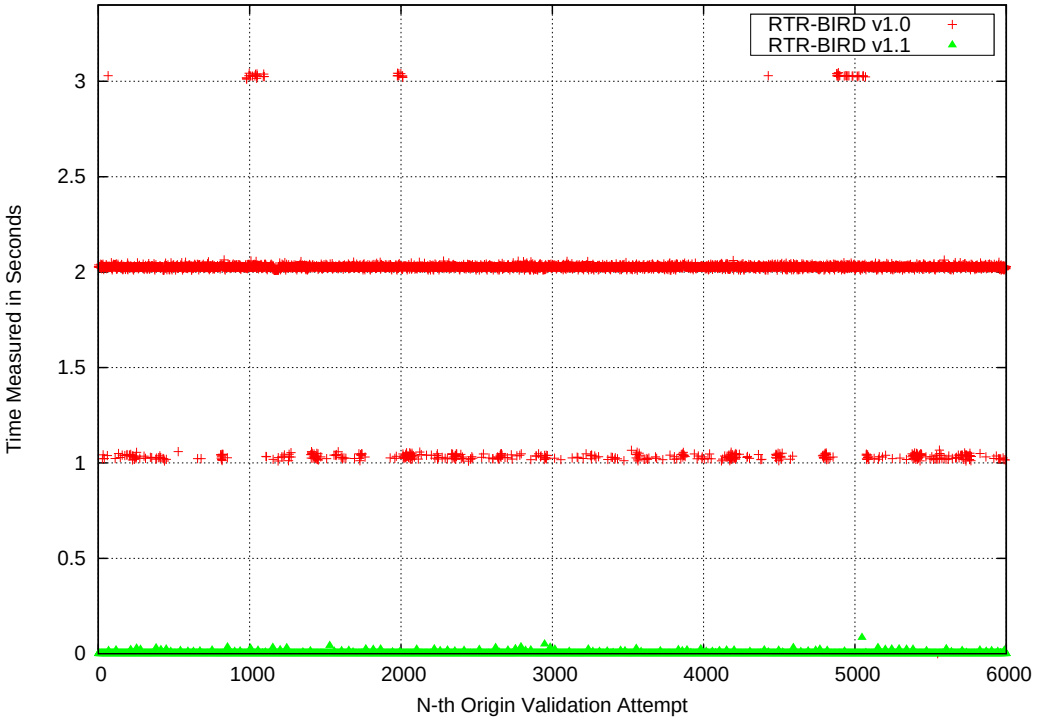


(b) Comparison in Accuracy between *RTRPKI* Variations

Figure 3-7: The Simulation Results of Different *Origin Validation Programs* (*OVPs*) and *RTR-BIRDs*

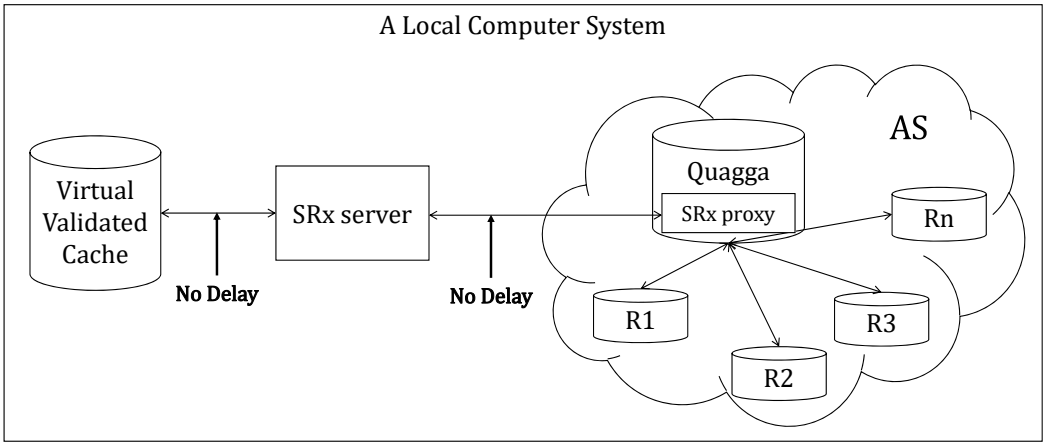


(a) *RTR-BIRD* v1.1 Versus Quagga*SRx* v0.3.1.1

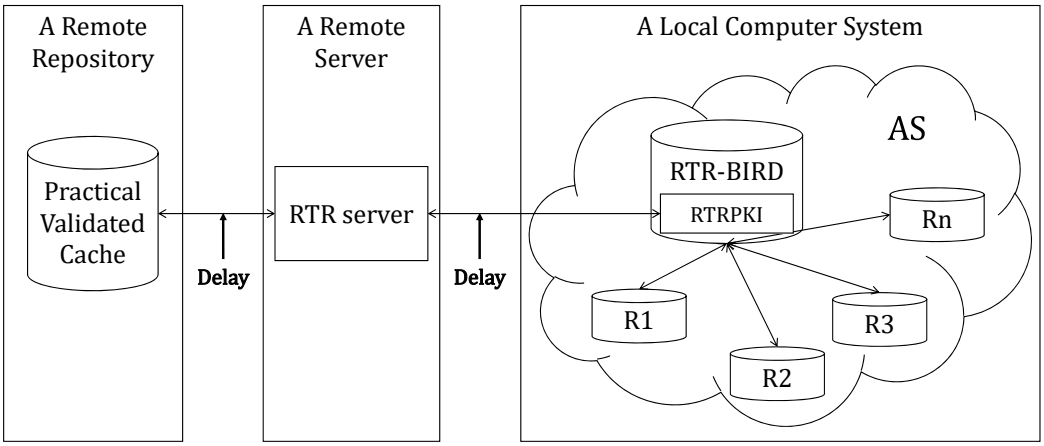


(b) *RTR-BIRD* v1.1 Versus *RTR-BIRD* v1.0

Figure 3-8: The Comparative Analysis in Performance between *RTR-BIRD* v1.0, *RTR-BIRD* v1.1, and Quagga*SRx* v0.3.1.1



(a) The Network Diagram of Quagga*SRx* v0.3.1.1 in Simulation



(b) The Network Diagram of *RTR-BIRD* v1.1 in Simulation

Figure 3-9: The Comparison of Network Diagrams Based on Each *RPKI*-capable Software Router

Chapter 4

The Feasibility Analysis of *BGPsec*

It is revealed in Chapter 3 that our implementation, *RTR-BIRD* v1.1, is able to "practically" validate origins of *BGP Update* messages without any additional delay. Both *Origin Validation* and *Path Validation* should be supported in order to entirely protect *BGP* from *BGP*-originated vulnerabilities as discussed in Section 3.1.3. Otherwise, numerous attacks can be launched by fabricating the *NLRI*s or *Path Attribute*s of *BGP Update* messages such as *injection*, *active intermediation*, *delay messages*, *replay attack*, *withholding attacks*, *saturation attacks*, and so forth [140]. The *RTR-BIRD* v1.1, however, supports only *Origin Validation*. The most plausible solution now is *BGPsec* to support both *Origin Validation* and *Path Validation*. However, *BGPsec* has been being drafted since 2011 over eleven times and has not yet been standardized [2]. Consequently, we analyze the reason for the procrastination of *BGPsec* standard in this chapter, in order to improve the security level of *RTR-BIRD* v1.1, as well as *BGP* networks at the end.

4.1 The Introduction to *BGPsec*

Amongst past proposals in Section 3.1, both *Origin Validation* and *Path Validation* are supported by *S-BGP*, *IRV*, *psBGP*, and *PGBGP*. Between them, *S-BGP* is the only solution providing a satisfiable security level as well

as infrastructure; this is detailed in Section 3.1.3. Our discussion here starts with *S-BGP* because *BGPsec* is constructed on the basis of *S-BGP* and *RPKI*. The *S-BGP* uses two types of route attestations that are *Address Attestation*s (*AA*s) and *Route Attestation*s (*RA*s) respectively for *Origin Validation* and *Path Validation*. An *AA* is issued by an organization that owns the *IP Prefix*es contained in the attestation. An *RA* is issued by a router authorized to announce the *ASN* that the router is included in. However, these attestations significantly increase the convergence time in *S-BGP* [141]. Moreover, the computational overhead in *S-BGP* needs the substantial resources of *CPU* and memory, and the storage requirements for *RA*s in *S-BGP* are also noted [113]. Although the convergence time and overhead in *S-BGP* can be argued as computer hardware evolves, another problem in *S-BGP* (as well as the other prior works) is that *S-BGP* requires a means of validating the following assertions with regard to the origination of a route into *BGP*; (i) the *IP Prefix* and *ASN* being used are valid, (ii) and the parties using the *IP Prefix* and *ASN* are properly authorized to do so in the context of the routing advertisement.

Resource Public Key Infrastructure (*RPKI*) [142], which is proposed by *Secure Inter-Domain Routing* Working Group (*SIDR* WG) of *Internet Engineering Task Force* (*IETF*), formalizes those assertions using *Resource Certificates* [143]. In *RPKI*, the independently constructed attestations in *S-BGP* (i.e., *AA*s) are substituted to *Route Origin Authorization*s (*ROA*s) [144], each of which is an object [145] signed by a resource holder owning an associated *End-Entity* (*EE*) certificate, which certifies an ownership (i.e., *ASN*) of one or more *IP Prefix*es. That is to say, the *S-BGP* assertions (i) and (ii) above are not required in *RPKI*. In other words, the validation of a *Signed Object* (e.g., a *ROA*) is not required as far as an associated *EE* certificate has not expired (i.e., "valid") and has existed; conversely, a *Signed Object* can be considered invalid if its associated *EE* certificate has been revoked. At this moment, approximately 20, 000 unique *IP Prefix*/origin pairs over 570, 000 are protected by *RPKI*, and the number of valid pairs is gradually being increased

[122, 131].

The biggest problem in *RPKI* is that it does not support *Path Validation*, and the importance of *Path Validation* is emphasized with an example in Figure 4-1. Note that all three types of business relationships (i.e., *Provider-to-Customer*, *Peer-to-Peer*, and *Customer-to-Provider*) and the *Gao-Rexford model* (*GR model*) are preserved in the example. After all routers are converged in the network, the *AS* path from $AS1$ to $AS5$ should be $AS1$-$AS2$-$AS4$-$AS5$ because another path violates the *GR model*. What if $AS3$ is a malicious router and wants to hijack the traffic? The $AS3$ can easily hijack the traffic to $AS5$ by announcing the *AS* path $AS3$-$AS5$ to $AS1$. If $AS3$ has a provider link to $AS4$, $AS3$ can "intercept" the traffic for more invisible and long-term attacks [37] rather than hijacking it. Speaking of which, the ***Prefix Interception*** is referred to hijack and forward traffic as expected, forming a *Man-In-The-Middle* (*MITM*) attack. In the case of ***Prefix Hijacking***, hijacked traffic does not reach an original destination; in other words, an *AS* can drop all traffic and give rise to a *Denial-Of-Service* (*DOS*) attack against the prefix owner [110]. The traffic also can be redirected to an incorrect destination and utilized for a phishing attack [36]. As far as the *AS_Path* attribute is not validated, *Prefix Hijacking* would be available in any way. In *BGP*, the *Communities* attribute [21] or *AS-Path Prepending* [1] may support *Path Validation* by importing or exporting legitimate validation information between *BGP* speakers but they are insufficient.

4.2 The Formulation of the Computational Overhead in *BGPsec*

BGPsec [146, 147], which is being developed by *IETF* network working group, extends *RPKI* by adding an additional type of certificate referred to as a *BGPsec Router Certificate*. A *BGPsec Router Certificate* binds an *AS* to a

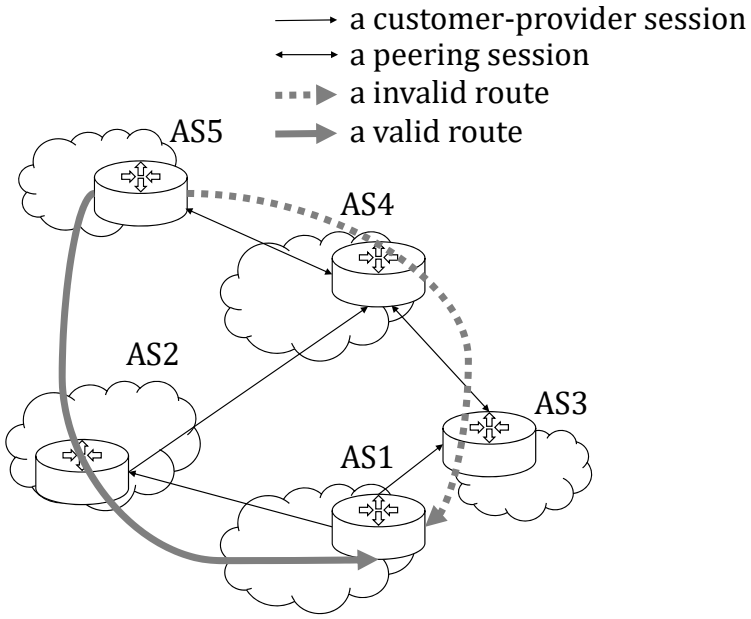


Figure 4-1: The Example of Possible *IP Prefix* Hijacking on *BGP*

public signature-verifying key, of which a corresponding private key is held by one or more legitimate *BGP* speakers included in the *AS*. A ***Relying Party (RP)*** (e.g., *rcynic*) retrieves *RPKI* or *BGPsec* objects (e.g., a *BGPsec Router Certificate*) from repositories (e.g., an *RPKI* validated cache) and actually performs *Origin Validation* or *Path Validation*; that is, it facilitates *BGPsec Router Certificate*s to verify *BGPsec* signatures in *BGPsec Update* messages produced by *BGPsec* speakers. Speaking of the *AS*-signing process in *BGPsec*, a *BGPsec* speaker signs its *ASN* and appends the signature in a *BGPsec Path Attribute* before propagating the message to its neighbors. On the receipt of a *BGPsec Update* message, a *BGPsec* speaker verifies all signatures in the *BGPsec Path Attribute* of the message through an associated *RP* in the order of last-to-origin *AS*. In doing so, *BGPsec* can protect the conventional *AS_Path* attribute in a *BGP Update* message [147]. Figure 4-2 demonstrates the format of *BGPsec Path Attribute* in octets, complying with the eighth draft of *BGPsec* specification. Each *BGPsec Path Attribute* consists of a *Secure-Path* attribute and a sequence of *Signature-Block*s.

Each signature is generated and added by an associated *BGPsec* speaker

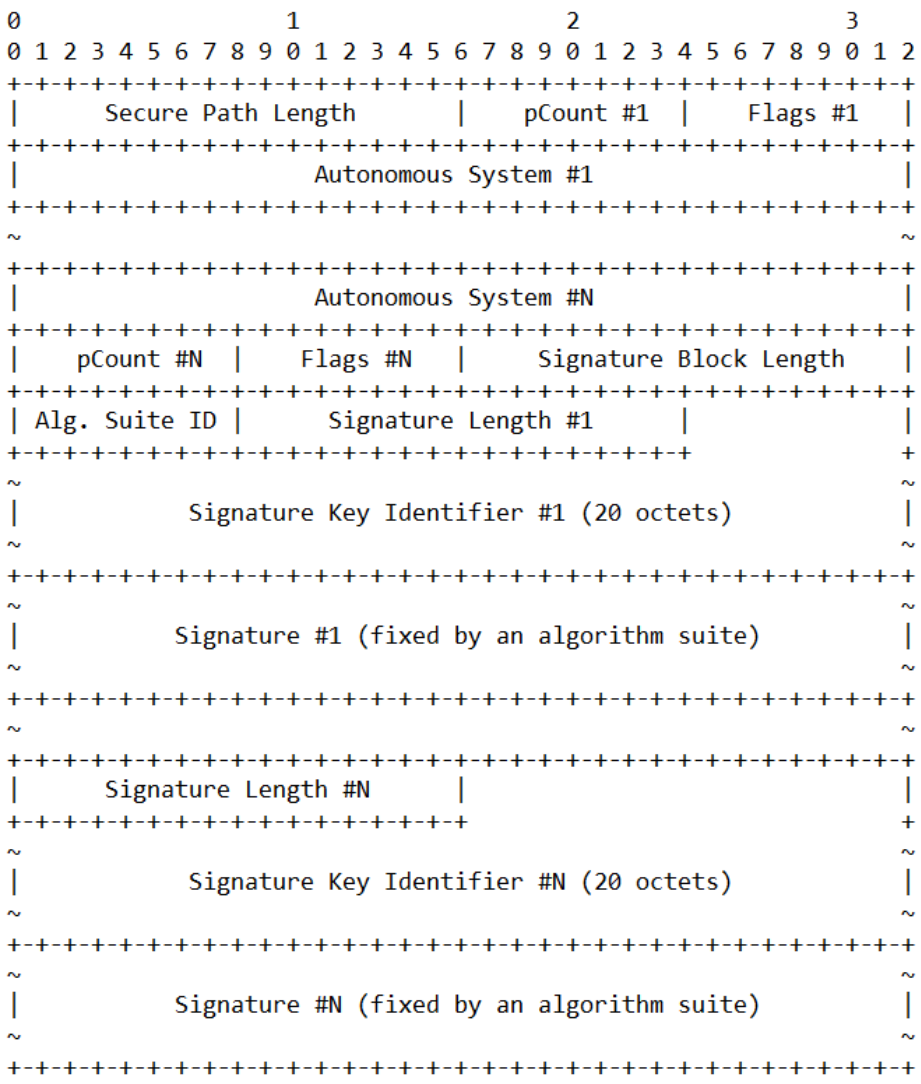


Figure 4-2: The Format of the *BGPsec Path Attribute* in a *BGPsec Update* Message [2]

into a *Signature-Block* in a sequence; that is, this signing process is performed at each speaker from an origin *AS* all the way to a destination *AS* specified in a *Secure-Path* attribute. Each signature is fastened with an *AS* in the *Secure-Path* attribute in a same *BGPsec Update* message, as a result, one or more pairs of a signature and a corresponding *ASN* constitute a *BGPsec Path Attribute* (*non-transitive*), which is the core part in *BGPsec*. Consequently,

we derive from the protocol specification [2]

$$\Bigg(\Big(N+m\Big)\Big(L_{cache}+T_{seek}+\big(\frac{N+m+1}{N+m}\big)T_{hash}\Big) + mT_{verorig}+T_{sign_{path}}+NT_{verpath}\Bigg)\Big(N-1\Big), \tag{4.1}$$

which is the computational overhead of *BGPsec* in milliseconds to validate signatures in a *BGPsec Path Attribute*. Note that this formula (i) exempts time consumed for a general *BGP*-routing procedure while processing a *BGPsec Update* message; for example, the latency for sending/receiving a *BGP Update* message or selecting the best path, (ii) assumes that the exchange of *BGP* capability advertisements, which are leveraged to check the *BGPsec* capability between *BGP* speakers, has been finished to use *BGPsec* in a specific address family (i.e., *IPv4* or *IPv6*) [148], (iii) and also presumes that the network delay in *BGP* is identical to that in *BGPsec* in the same *AS* path; actually, *BGPsec* shows a larger packet size than *BGP* due to the *BGPsec Path Attribute*.

N is the number of *AS*es in a given path. *ROA*s are generated through outbound traffic as such it can be excluded from the overhead, whereas *BGPsec Path Attribute*s are created using inbound traffic at the time of re-advertisement [142]. Each *BGPsec* speaker acting as a sender should create its signature to propagate a path, resulting in $T_{hash}+T_{sign_{path}}$; subsequently, the total overhead to generate a *BGPsec Update* message from an origin to a destination is $(N-1)(T_{hash}+T_{sign_{path}})$. Note that, when a *BGPsec* speaker generates a *BGPsec Path Attribute*, it uses its own private key so there is no additional network delay. T_{hash} is a delay to create an *SHA*-256 digest in *BGPsec* [149]. $T_{sign_{path}}$ is time to sign a given octet sequence [2]. The current version of *BGPsec* employs two signature algorithms. The first algorithm is *RSASSA-PKCS1-v1_5* [150] for certificates, *Certification Revocation List*s (*CRL*s), and *Signed Object*s. The second one is *Elliptic Curve Digital Signature Algorithm* (*ECDSA*) for certification requests and *BGPsec*

Update messages [151]. Note that the minimum size of an *RSA* public key in *RSASSA-PKCS1-v1_5* is 2048 bits, and the curve used for *ECDSA* in *BGPsec* is *secp256r1*. In other words, the signature algorithm used in the $T_{sign_{path}}$ in Equation (4.1) is *ECDSA*. On the other hand, each speaker acting as a receiver should perform verifications for both *NLRI* and *AS* paths. Every *BGP Update* for the announcement has one or more *IP Prefix*es as *NLRI* so each of the speakers should validate m origins where m is the number of *IP Prefix*es. As a consequence, the total delay to validate all *IP Prefix*es by all message receivers (i.e., *BGPsec* speakers) is $m(N-1)(L_{cache}+T_{seek}+T_{hash}+T_{verorig})$. All verifications in *BGPsec* require public keys of remote *AS*es so they produce the network delay, $L_{cache}+T_{seek}$, which is time to request an *EE* certificate to an *RPKI* validated cache and time to search/return the corresponding information to the requester *AS* between the *AS*, an *RP* (i.e., a cache server), and a validated cache. Note that $T_{verorig}$ here is time to verify an *RSA* signature. For *Path Validation*, each *BGPsec* speaker on a path needs to verify $(x-1)$ signatures where x is the total number of *AS*es (or the total number of *eBGP* speakers) on the perspective of a receiver *AS*; for example, if a *BGPsec* speaker in *AS*4 receives a *BGPsec Update* message with the *BGPsec Path Attribute*, *AS*1-*AS*2-*AS*3, x will be four including the self-*ASN* (i.e., *AS*4); on the next *eBGP* speaker, x will be intuitively five. Therefore, the total number of verifications for the *BGPsec Path Attribute* of a *BGPsec Update* message is $2\sum_{x=1}^{N-1} x$; in detail, the coexistence of two algorithm suites [2] doubles the number of verifications. As a result, we arrive at the expression, $2\sum_{x=1}^{N-1} x(L_{cache}+T_{seek}+T_{hash}+T_{verpath})$, which is time to verify all signatures for a *BGPsec Path Attribute*. Equation (4.1) is a simplified form of the three equations discussed in this section.

4.3 The Extraction of Practical Measurements from the *BGP* Formula

In order to "measure" the practical values of variables in Equation (4.1), we wanted to facilitate a *BGPsec* software router to "calculate" the result of the equation. At the time of our study, Quagga*SRx* [136] implemented by *NIST* was only *BGPsec* software router but it had leveraged virtual key pairs; in the *BGPsec* network based on the Quagga*SRx*, *PKI* key pairs were self-signed and stored in a local file. In other words, $(N^2 + (m-1)N - m)(L_{cache} + T_{seek})$ will be zero in Equation (4.1). Because of that, we have selected *RTR-BIRD* v1.1 [152] instead, which is our software router implementation supporting *RPKI*; in specific, it does not support *BGPsec*. The *RTR-BIRD* v1.1 merely produces $(L_{cache} + T_{seek})$ in *Origin Validation*. Speaking of which, when a *BGP Update* comes into the router, (i) the implementation requests a validation to a corresponding cache server, for which we utilized *RIPE NCC RPKI Validator* version 2.16, (ii) and then it determines to update its routing table (i.e., *Loc-RIB*) depending on the validation result obtained from the server. To retrieve the practical value of $(L_{cache} + T_{seek})$ using *RTR-BIRD* v1.1, we implemented two more additional programs; a *BGP Update* generator and an *RTR-BIRD* timestamper. A *BGP Update* generator creates *BGP Update* messages and sends them out to peers; for our simulation, 8000 *BGP Update* messages were created based on 8000 valid *ROA*s retrieved from an *RP* (i.e., a *RIPE NCC RPKI Validator*) and were sent out to the *RTR-BIRD* v1.1 ten times via a *TCP* connection so they all should be considered valid in *Origin Validation*. An *RTR-BIRD* timestamper logs the *Origin Validation* overhead whenever a *BGP Update* message is received and processed by *RTR-BIRD* v1.1. Note that all software was installed on different computer systems in a laboratory and connected through the *Local Area Network* (*LAN*),

and each computer system was operated on *Ubuntu 12.04.5 LTS* with a single 3-GHz *CPU*.

Figure 4-3 visualizes the results of this simulation. Almost half of the *BGP Update* messages are validated within 20 microseconds, and the second most significant group in overhead is around 60 microseconds around which approximately 10, 000 *BGP Update* messages are origin-validated. The other *BGP Update* messages are almost randomly validated as shown in Fig. 4-3. We found from the simulation that a *BGP Update* message is validated by *RTR-BIRD* v1.1 in 0.6 milliseconds, and this is $(L_{cache} + T_{seek})$. It is because, we had removed all other factors from the original computational overhead formula, which is $m(N-1)(L_{cache}+T_{seek}+T_{hash}+T_{ver\,orig})$ in Equation (4.1), as follows. (i) We put a single *NLRI* into every single *BGP Update* message used in our simulation so $m = 1$, (ii) Those *BGP Update* messages originate from the *BGP Update* generator, which is directly connected to the *RTR-BIRD* v1.1 on the experiment, so $N = 2$. (iii) In *RPKI*, a *BGP* speaker can selectively validate a *BGP Update* message [144]; in specific, an associated cache server can pre-validate pairs of origin information as a proxy and just provide the validation result when it receives a validation request. In doing so, the delay, $T_{hash}+T_{ver\,orig}$, is removed as well in our simulation. In addition, the *OPENSSL* version 1.0.1f is leveraged to measure the values of $T_{sign_{path}}$, $T_{ver\,orig}$, and $T_{ver_{path}}$, in Equation (4.1). On a single 3-GHz *CPU*, the *secp256r1* curve (a.k.a. *nistp256*) takes 0.1 and 0.4 in milliseconds respectively for signing and verifying a signature, whereas *RSA* with a 2048-bit public key takes 1.4 and 0.04 in milliseconds. Incidentally, *SHA*-256 takes up to six nanoseconds to process under ten kilobytes. The total number of unique *IPv4* prefix/origin pairs is approximately 570, 000 [131] (*IPv6* figure is 25, 000 [153]), and the total number of *AS*es is around 69, 000 [154] on a global scale. Thus, an *AS* owns eight prefixes in average, and the average *AS_Path* length is four [122].

Table 4.1: The Probability Distribution Table Categorized by the Number of *AS*es in the *Path Attribute*s of All *BGP Update* Messages Accessed on March 17, 2014

Probability Distribution																Total
The Number of *AS*es	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	44 105
The Number of Paths	2	22	1066	2277	1726	1404	623	326	153	185	111	56	30	45	106	8132
Probability in Percent (%)	0.02	0.27	13.10	28.0	21.22	17.27	7.66	4.0	1.88	2.27	1.36	0.69	0.37	0.55	1.30	100

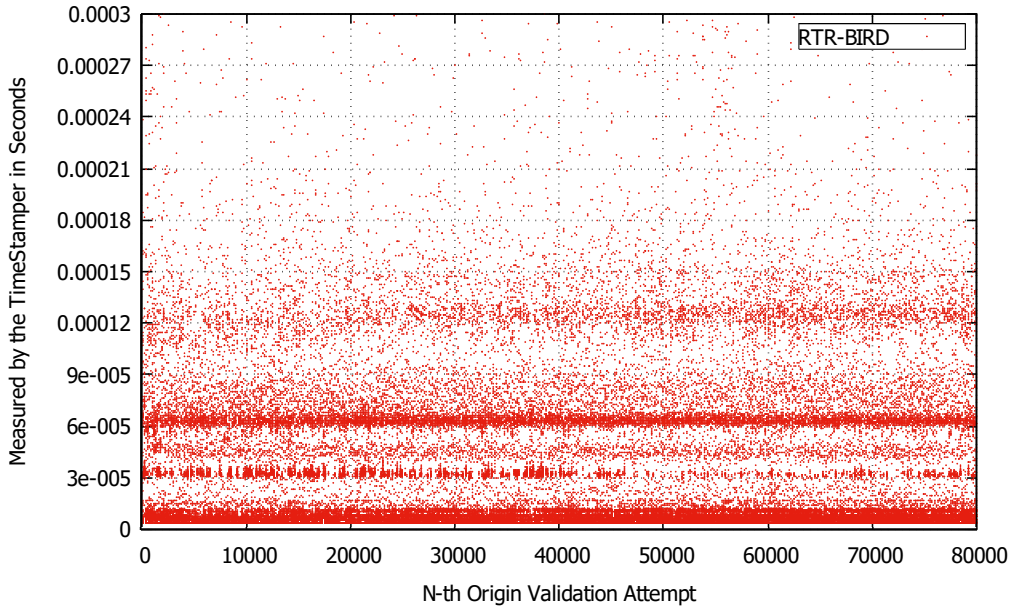
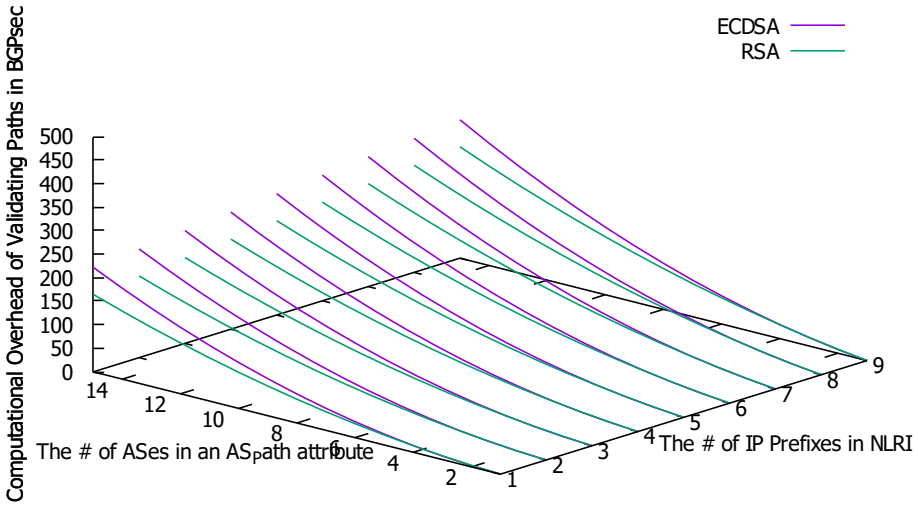


Figure 4-3: The Network Delay in Seconds of Each *Origin Validation* of More than 80, 000 *BGP Update* messages on *RTR-BIRD* v1.1

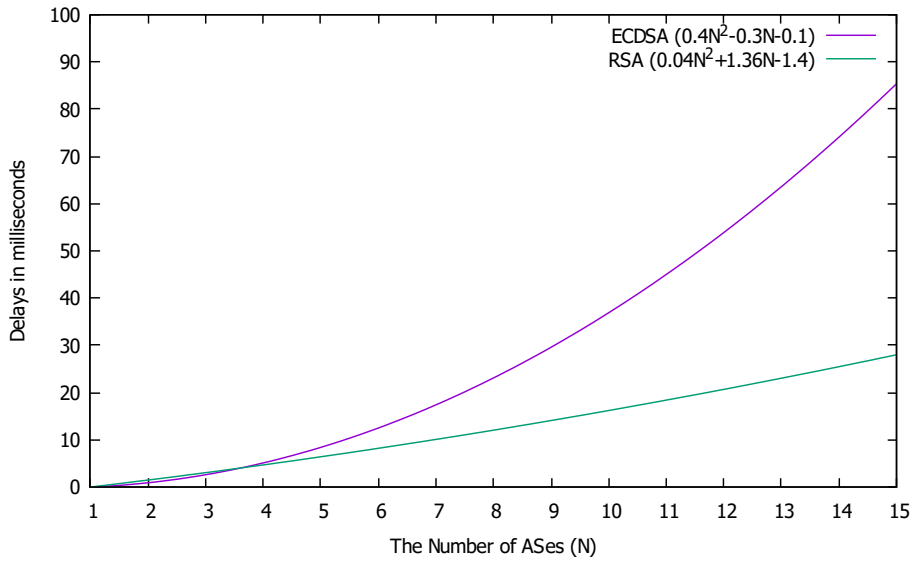
4.4 The Analytical Results of Equation (4.1) with the Practical Measurements Derived in Section 4.3

By substituting all variables in Equation (4.1) to the practical measurements acquired in Section 4.3, we reach at about 27.66 milliseconds, which is the additional computational overhead "in average" imposed to *BGPsec* for both *Origin Validation* and *Path Validation*. The *RTR-BIRD* v1.1 generally processes a *BGP Update* in 0.4 milliseconds including *Origin Validation*. Namely, the *BGPsec* computational overhead decreases the *BGP* routing

performance approximately 69.15 times in average. In order to lessen the overhead, we here propose to facilitate *RSA* for both validations. The average overhead when using only *RSA* in *BGPsec* is approximately 27.24 milliseconds that is pretty similar to the case of using *ECDSA* to validate paths in *BGPsec*; in the "average" case, although verifying a path in our proposed structure is ten times faster than that in the original *BGPsec*, signing a path offsets the difference between them. However, Equation (4.1) shows its further impact on *BGPsec* routing performance. The remarkable part of the equation is the time complexity of $T_{sign_{path}}$, which is $O(N)$ while the term $T_{ver\,path}$ shows $O(N^2)$ in time complexity. So as an *AS* path grows in length, the computational overhead of current *BGPsec* protocol will be exponentially increased. Figure 4-4a illustrates the computational overhead of validating paths using *ECDSA* or *RSA* in *BGPsec*, depending on the *AS* path length and the number of *IP Prefix*es in a *BGPsec Update* message. The computational difference is emphasized in Figure 4-4b. By that means, we analyzed an archive file of *RouteViews* [5] in a *Multi-threaded Routing Toolkit* (*MRT*) format [155] to see the probability distribution of all *AS*es. According to the data shown in Table 4.1, *RSA* introduces better performance in 58.61 percent.



(a) The Computational Overhead in Milliseconds Depending on the Number of *AS*es and *IP Prefix*es in a *BGPsec Update* Message



(b) The Computational Overhead in Milliseconds Depending on the Number of *AS*es For More Visibility in Comparison

Figure 4-4: The Calculated Computational Overhead of BGPsec Based on *ECDSA* and *RSA*

Chapter 5

BGP Data Analysis: What are Insecure and How to Protect Them?

So far, protocol-based studies have been introduced in Chapter 3 and 4. For *Origin Validation* in *BGP*, *RPKI* is discussed and implemented as a software router called *RTR-BIRD* v1.1. For *Path Validation* in *BGP*, *BGPsec* is analyzed on the basis of its protocol specification and updated in its path validation for better performance. However, the tie-breaking rules leveraged in *BGP* is highly dependent on *BGP* policies of each *AS* as discussed in Section 1.2.6 so actual routing behaviors can be different from what we expected. In addition, our laboratory does not have hardware resources that can be facilitated to simulate those kinds of real-world *BGP*-networking. This physical barrier leads us to the *BGP* data analysis in this chapter.

5.1 The Motivation to the *BDC* Development

BGP networks could be accomplished by a network simulator such as *GNS3* [156] or *ns-2* [157]. Those simulators are featured in comfortably depicting *BGP* networks and configuring *BGP* routers in very detail. The *BGP* network originating from those simulators are, however, unsuitable for the challenging experiments such as the application of a new solution (e.g., *BGPsec*) because the modification of infrastructure will be required in these observations as in

Section 4 but is not supported by those simulators. Instead, the environment installation of *BGP* network can be achieved by an open source software router such as the *BIRD* [133] or *Quagga* [158]. In specific, by revising a software router in structure, researchers can fulfill the flexible environment installation. This is the reason of our studies in Chapter 3. At this moment, more variations of *BIRD* and *Quagga* are also available [136, 159].

After installing an experimental network, it is significant to import *BGP* data from a real-world *BGP* speaker. In other words, at least, one speaker in the exploratory network should establish a *BGP* session to a real-world *BGP* speaker, in order to attain real-world data into the network. It is, however, difficult for researchers to have the session connected for a simulated purpose because the interactive configuration for a *BGP* session is required and will be offered mostly for revenue. In other words, the virtual or purposeless information will be running around and leveraged by the network, unless the real-world *BGP* information is available. Fortunately, the *RouteViews* project [5] provides the real-world *BGP* information on a global scale, and many studies on *BGP* rely on the information due to the reasons discussed. For example, *BGPlay* [160] demonstrates animated graphs of the *BGP* routing activity of a certain prefix, within a specified time interval. *BGPmon* [161] observes *BGP* routing data and produces the live stream of *BGP Update* messages, which can be retrieved via a *TCP* connection. Speaking of which, Hu and Chen [162] focus on the cost-effectiveness of *BGPmon* and show that much smaller number of *Vantage Point*s can retain the original monitoring capability in the monitor. The past data [5] collected by *RouteViews* is also accessible on the *RouteViews* website in an *MRT* [155] format. The *Classless Inter-Domain Routing* (*CIDR*) report [163] summarizes the total routing table size, route aggregation, *Bogon*s, and so on. These reports are created also based on the forwarding and routing table entries of *AS*6447 (The university of Oregon) in a daily fashion. The *Center for Applied Internet Data Analysis* (*CAIDA*) contributes on the quantitative ranking between *AS*es or organizations (i.e.,

each of which consists of one or more *AS*es) [164] and offers the basic lookup for an *AS* or organization. Similarly, the *Hurricane Electric* (*HE*) [153] provides various reports about prefixes, peers, *Bogon*s, multiple origins, *Domain Name Server*s (*DNS*s), and so on. More implementations [154, 165] do exist but they just provide basic *BGP* information (e.g., *IP* statistics, *ASN* lookup) in a similar way. The most useful tool to analyze *BGP* data would be *REpresentational State Transfer Application Programming Interface* (*REST API*) supported by *RIPE NCC*, so called *RIPEstat DATA API* [166]. The *RIPEstat DATA API* offers the selective *BGP* information in a timely manner. Upon users' *Hyper Text Transfer Protocol* (*HTTP*) request, *BGP* information will be selectively extracted from the *RIPEstat* database. For example, a user can obtain the announced prefixes or the information of network neighbors for a given *ASN* in a historical fashion.

However, those statistical data is insufficient to inspect the behavior of a *BGP* network in routing or security. For example, network administrators may want to know which *AS* path is suspected to be invalid. As another example, they may want to know how many paths can be possibly accepted into the *Loc-RIB* of a *BGP* speaker; for example, the paths not complying with the *GR model* should be considered invalid. The data-analyzing tool to answer those questions does not exist at this moment. As more realistic data-analyzing studies, Wu et al. [167] present the design and evaluation of an online system that converts millions of *BGP Update* messages a day into a few dozens of reports about significant routing disruptions. Sriram et al. [168] present an overview and comparisons of *BGP* robustness and anomaly detection algorithms. They review some algorithms and alert tools such as the *Nemecis* tool, *PGBGP*, *PHAS*, et cetera. They also propose a new algorithm and a new evaluation methodology called *TERRAIN* on which those algorithms are being tested and empirically compared. Gregori et al. [169] propose a technique to geolocate the *AS* connections retrieved from *BGP* raw data, in order to highlight the *Internet* regional characteristics.

Their results show some regional characteristics in terms of graph properties and inter-*AS* economic relationships. To the best of our knowledge, *ExaBGP* and *Trac* are smartest ones among all previous solutions. Speaking of which, *ExaBGP* [170] is designed to gather network information, mitigate network attacks, and provide *failover* solutions; in addition, this implementation fully follows the *BGP* Finite State Machine (FSM) so that it can configure *BGP* and legitimately send out *BGP Update* messages. *Trac* [171] provides comprehensive *RPKI* components such as *Certification Engine*, *Relying Party (RP) Cache* (e.g., *RIPE NCC RPKI Validator*), *RPKI*/RTR protocol, and related web-based reports.

5.2 The Implementation of *BGP Data Center* (*BDC*)

The best way to analyze "actual" routing behavior without network simulation is to intuitively look into routing data, that is, incoming *BGP Update* messages and *RIB* entries (i.e., *Loc-RIB* entries) in *BGP* networks. Otherwise, it is almost impossible to accurately investigate *BGP*-routing behaviors due to the many factors given in *BPSA*, as well as routing policies. Up until now, the *BGP* data-analyzing tools introduced by researchers just provide raw *BGP* data, which is insufficient to locate unexpected routing problems. Therefore, we here propose an all-in-one solution to retrieve wisdom from those data or information. Basically, our solution called the *BGP Data Center* (*BDC*) presents analytical results by examining *BGP* data, which can be either *BGP Update* messages or *RIB* entries. The *BDC* at present is able to (i) crawl *BGP* data from *RouteViews*, *Réseaux IP Européens Network Coordination Centre* (*RIPE NCC*), or *BGPmon*, (ii) verify *NLRI*, (iii) convert a data format between JavaScript Object Notation (*JSON*), eXtensible Markup Language (*XML*), and a *BGP* packet, (iv) produce outbound *BGP* traffic,

(v) and offer *BGP* data-analyzing functions. As a consequence, *BDC* provides comprehensive functional supports of the *Data-Link Layer* and *Network Layer* in *Open Systems Interconnection* (*OSI*) model so that it can be ultimately used to yield the diversity of data-investigating capability, as well as the correction of *BGP* updates on a *BGP* session. Incidentally, the modularized and structured design of *BDC* will allow users to construct their data-inspecting recipe upon their taste.

5.2.1 The System Requirements of *BDC*

Basically, the first requirement for an all-in-one data analyzer is to gather real-world *BGP* data in a flexible manner. Our implementation should support a selective data-crawling operation to be differentiated from the other developments; *BDC* gathers real-world *BGP* data from various sources as elaborated in Section 5.2. The next requirement is to approach more practical problems by analyzing the collected information. The indispensable feature here is to offer various ways to "communicate" with our implementation as much as we can, rather than to just present instructions for the limited number of analysis so that the number of solvable data-analyzing topics in the implementation can be maximized. Thereby, we facilitate the *Command Line Interface* (*CLI*) to support an interactive communication within our system. In particular, *BDC* introduces the data-analyzing chunks rather than a complete one. These pieces minimize the dependency between data and functions, as well as yield the flexibility in the analysis. As a consequence, it is able to support the numerous number of problems by manipulating the finite set of operations. In addition, the *BDC* adopts the client-server model for the case when the process of our development is reserved for a single task for a while; for example, when acquiring real-world *BGP* data from the *RouteViews* (*AS*6447) or *BGPmon*. Otherwise, a user requires waiting until the process is terminated. To be specific, any operation that provides an instant response to a user is situated on the client side, whereas laborious functions reside on

Instruction Type	Sniffing Type	Sniffing Source	Sniffing Destination
Crawling Type	Selective Crawling	Crawling Target ID	Crawling Value

Figure 5-1: The Specification of *BDC* Protocol

the server side; this arrangement brings out the *BDC* protocol as defined in Figure 5-1.

In particular, time-consuming functions are the *Crawler* and *Sniffer* to crawl *BGP* data and sniff *TCP* sequence numbers on a *BGP* session. By that means, the controls for those functionalities are incorporated into the *BDC* protocol to manage the corresponding operations on a server side. Last, the *BGP* data retrieved from different sources needs to be unified in the format before proceeding to data analysis. In other words, the conversion of data types between the *RouteViews*, *BGPmon*, and *Réseaux IP Européens Network Coordination Centre* (*RIPE NCC*), should be supported.

5.2.2 The Structural Explanation of *BDC*

The comprehensive *BGP* data-analyzing solution, which satisfies the requirements discussed in Section 5.2.1, is constructed as depicted in Figure 5-2, on the basis of Python 2.7.6. The implementation is divided into four major components; that is, the *Operating Module*, *Crawling Module*, *Analyzing Module*, and *Auxiliary Module*. Each subsection will describe each module.

The *BDC Operating Module*

The *Operating Module*, which consists of *CLI*, *Shared Library*, and *File Formatter*, provides a construction base of *BDC* to the other modules as an operating system does to programs in a regular computer system. The *Command Line Interface* (*CLI*) module is mainly featured in (i) offering user-usable functions by importing base functions from the other modules and making the defined functions available through the *CLI* interface,

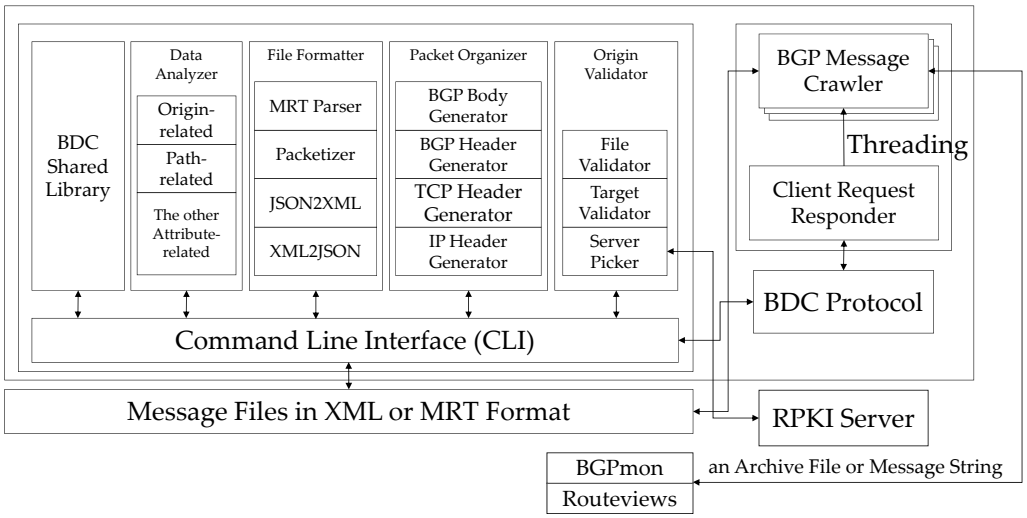


Figure 5-2: The Structural Design of a Comprehensive Data-analyzing Tool called *BGP Data Center* (*BDC*)

(ii) providing *CLI*-based functions (e.g., auto-completing a command, offering '–help'), (iii) managing intermediate information in the analysis. The *Shared Library* module mostly defines sharable functions between modules (i.e., module-independent functions), which includes (i) simple array functions for look-up, sorting, matching, and so forth, (ii) data-parsing functions, (iii) managing intermediate information of analysis, (iv) et cetera. The *File Formatter* module provides all format-converting functions between *JSON*, *XML*, and *MRT*, as well as *Packetizer* using which network data can be packed or unpacked into or from an *IP* packet.

The *BDC Crawling Module*

The *Crawling Module* is able to retrieve *BGP* data from *RouteViews*, *BGPmon*, and *RIPE NCC*. Each of the sources necessitates a different type of inquiry. In specific, *RouteViews* (*AS*6447) data is attainable from archive files published on their website. All *BGP Update*s and *RIB* entries since 2001 have been maintained by *RouteViews* as archive files, each of which demonstrates data in an *MRT* format respectively for *BGP Update* or *Table Transfer* messages. Our *Crawling Module* is distinguished from the other

data-collecting tools in a selectable download in terms of a period or naming convention; for example, this module can request the download of *BGP* data between 00:00 and 00:15 on January 2016, every February between 2011 and 2016, and so on.

Another feature of our implementation is to gather *BGP* data from *BGPmon*, which provides a *TCP* live stream of *BGPmon* update messages in an *XML* format. Particularly, the live stream is stored into *BDC* in an *XML* or *JSON* format in a real-time fashion through a *TCP* connection to the *BGPmon* service. The data type is called *BGP_MONITOR_*MESSAGE, which is either a *BGP Update* message or an *RIB* entry; in other words, users can select a data type corresponding to their needs. For instance, if *BDC* users want to observe the *AS* paths of *AS*6447 to a certain *AS*, they can select *RIB* entries to be collected. If they want to know the *AS*es who are insisting (i.e., announcing) the ownership of a certain *IP Prefix* to *AS*6447, then they will require incoming *BGP* updates. If what they want to study is the policies of an *AS* (i.e., *AS*6447), both types of data will be required. The *Crawling Module* recognizes each *BGP Update* message (i.e., *BGP_MONITOR_*MESSAGE) from the *BGPmon* stream. Basically, the stream is a continuous string of *BGP_MONITOR_*MESSAGE elements in *XML*. *BDC* retrieves complete messages from the string and simultaneously documents them into a file; in other words, incomplete *BGP_MONITOR_*MESSAGE elements will be dismissed. In addition, users can claim a whitelist by a command for a selective collection; for example, users can obtain the messages that include a certain *ASN* in the *Path Attribute* or that originate from a certain *ASN*. The *RIPE NCC* is the last source of *BGP* data that can be crawled by *Crawling Module*. *RIPE NCC* provides their information upon a *Hyper Text Transfer Protocol* (*HTTP*) request through their *REpresentational State Transfer* (*REST*) *Application Programming Interface* (*API*) [166], resulting in a *JSON*-formatted data as a response. The advantage of the *RIPE NCC* data *API* is that various and valuable assets are acquirable from this *RIR*;

for example, *ASN* Neighbors, Announced Prefixes, Routing History, Country Resource Stats, et cetera [166].

Due to the three different types of data from three distinct origins, the *Crawling Module* maintains three intermediate variables and files for data analysis. The reason for preserving data in both a variable and file is to prevent the duplicated data-crawling request to a *BDC* server. Speaking of which, a *BDC* server will provide a lastly created file, without actually regathering data, to the client who sends an inquiry that is twinned to the *BGP* data request of another client. Among *MRT*, *XML*, and *JSON*, we choose a *JSON* format as a default format in our system, and the decision is based on the measurements illustrated in Table 5.1. In specific, a recently published archive provides by *RouteViews* introduced around 66, 000 *BGP Update* messages and approximately 440, 000 *RIB* entries respectively for *BGP Update* and *RIB* dumps. In the meantime, the *BGPmon* stream offers 15, 796 *BGP_MONITOR_*MESSAGE elements that were stored in an *XML*- and *JSON*-formatted file. The *Data Access* field designates the time in seconds to count the total number of *BGP Update* messages in the *MRT*, *XML*, and *JSON* file. The *Parsing Time* field indicates the time to load and parse those files into a *BDC* client. In detail, the access time for an entry is 155, 0.249, and 0.279 microseconds in an *MRT*, *XML*, and *JSON* format. In addition, an entry in each type is parsed respectively in 0.0015, 3200, and 160 microseconds in Table 5.1. These numbers are derived from 3.1-GHz *CPU* and four gigabytes of *RAM*. Although the *MRT* format is featured in the super-fast data-parsing due to its as-is exportation idea [172], this benefit will be leveraged once in a *BDC* client. On the other hand, the *Data Access* will be performed in a *BDC* client as many as the number of *BGP Update* messages. In consequence, *MRT* cannot be chosen as a default format in *BDC* because of its procrastination in approaching data in spite of its stunning *Parsing Time*. In terms of the *Parsing Time*, *JSON* is finally selected as a default format in *BDC* because of the hesitation of *XML* in parsing data.

Table 5.1: The Comparison of *BGP* Data Formats in Performance to Search Entries and Parse Data

Data Format	The Number of *BGP* Updates	*Data Access* (in seconds)	*Parsing Time* (in seconds)
MRT	66, 289	10.2475	0.0001
XML	15, 796	0.0039	50.1089
JSON	15, 796	0.0044	2.5312

Incidentally, the delay to access data in an *MRT* format seems to originate from the difference in data fields compared to the other types. Table 5.2 demonstrates all fields that can be attainable through each data type. Simply speaking, whereas *XML*-formatted *BGPmon* streams supply data of only source and destination information (i.e., *IP* addresses and *TCP* ports) and *BGP* messages (i.e., *BGP Open*, *BGP Update*, *BGP Keepalive*, and *BGP Notification* messages) likewise to *JSON*-formatted one, *MRT* supports more extensive coverage because this format is designed for "routing" protocol transactions, rather than just for *BGP*; for example, *Open Shortest Path First* (*OSPF*), Table Dump, *BGP* messages, or *Intermediate System to Intermediate System* (*IS-IS*). This flexibility of *MRT* and its data-exporting concept, which is to replicate and maintain the original data and its structure, produce some extra fields such as the *Common Header*, *Extended Timestamp Header*, and *BGP Header*. Those three headers precipitate the additional fields such as the *MRT* Timestamp, *MRT* Type, *MRT* Subtype, *MRT* Length, *MRT*-extended Microsecond Timestamp, *BGP* Marker, *BGP* Length, and *BGP* Type in Table 5.2. These attributes are, however, redundant in *BGP* data analysis. On the other hand, *XML*-formatted and *JSON*-formatted *BGP* data involve only meaningful information for *BGP* data analysis. Also, the original octets in the packet of an original *BGP* message are also reachable in these types of data for backward compatibility. In other words, although these data are reduced in size, they are summarized and provide all mandatory information for data analysis, especially in withdrawing or depositing *BGP* origins and routes.

Table 5.2: The Accessible Fields of *BGP Update* Data from *RouteViews* (*MRT*) and *BGPmon* (*XML* or *JSON*)

MRT*-formatted *BGP Update	***XML*- or *JSON*-formatted *BGP Update***
Timestamp	Timestamp and Datetime
Type	N/A
Subtype	N/A
Length	N/A
Microsecond Timestamp	N/A
Type	N/A
Peer AS Number	Destination ASN
Local AS Number	Source ASN
Interface Index	N/A
N/A	COLLECTION
Address Family	Address Family Identifier (AFI)
Peer IP Address	Destination Address
Local IP Address	Source Address
Marker	N/A
Length	N/A
Type	N/A
Withdrawn Routes Length	N/A
Withdrawn Routes	bgp:WITHDRAW
Total Path Attribute Length	N/A
Path Attribute Flags	N/A
Path Attribute Type	N/A
Path Attribute Length	N/A
Origin Type	bgp:Origin
AS Path	bgp:AS_PATH
Next Hop	bgp:NEXT_HOP
Multi Exit Discriminator	bgp:MULTI_EXIT_DISC
Local Preference	N/A
N/A	bgp:ATOMIC_AGGREGATE
Aggregator	bgp:AGGREGATOR
Community	bgp:COMMUNITY
Originator ID	N/A
Cluster List	N/A
MP Reach NLRI	N/A
MP Unreach NLRI	N/A
Extended Communities	N/A
AS4 Path	N/A
AS4 Aggregator	N/A
AIGP	N/A
ATTR SET	N/A
NLRI	bgp:NLRI
N/A	OCTET_MESSAGE
N/A	METADATA

The *Analyzing Module* in *BDC*

In general, *BGP* routing data such as *BGP Update* or *Table Transfer* messages are issued by *RouteViews* or *BGPmon*. These data is facilitated in *BDC* as base data. We found in Table 5.2 that the difference in available assets between those two sources does not yield any distinction in data analysis; that is, all necessary factors for data analysis are provided by either type. The *RIPE NCC* yields *BGP Update* messages as well as diverse supplementary data such as the ownership of *IP Prefix*es or neighboring information in *BGP* networks [173]. The *Analyzing Module* is connected to all of these three data providers for *BGP* routing and supplementary data. This module is mainly leveraged to retrieve (i) one or more packets (i.e., a *BGP Update* or *Table Transfer* message), (ii) statistical/historical routing information, (iii) or other auxiliary data given by *RIPE NCC*, so as to produce more valuable information; for instance, *BDC* is utilized in Section 6 to extenuate the insecure path problem in *BGP* networks that leaves *BGP* path information unsafe.

In order to diminish the data-analyzing restrictions discussed in Section 5.2.1, the *Analyzing Module* additionally provides independence to instructions in a *BDC* client. Speaking of which, the client system records the response of each inquiry, based on which users can initiate a new inquiry. This cascading operation scheme is the key feature of *Analyzing Module* to extricate the restrictions caused by the finite number of instructions. The two-dimensional operating scheme revealed in Figure 5-3 is designed to accomplish the cascading functionality. In other words, we concentrated on cooperating instructions rather than solving a particular *BGP* problem at the time of *BDC* development.

All instructions in a *BDC* client are categorized into the basic or advanced operations. The base operations participate in managing *BGP* data (i.e., *Set*, *Load*, *Show*, and *Convert*) or in communication (i.e., *Send*), and the data generated by a base command can be facilitated by an advanced instruction.

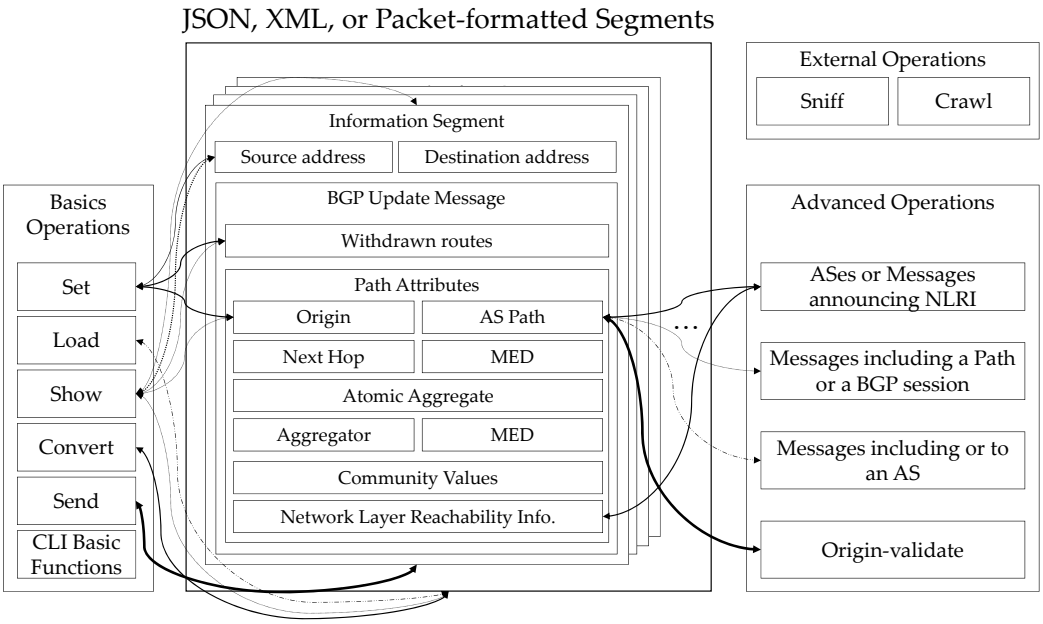


Figure 5-3: The Abstraction of Two-Dimensional Operations in *BDC*

For example, the advanced operation leverages the data to detect the *BGP Update* messages (i) that have the *NLRI* owning a given *IP* address or prefix, (ii) that include a stated *ASN* in their *Path Attribute*s, (iii) or that possess the shortest path to a specified *AS*. Subsequently, this information is collaborated with the auxiliary information of *RIPE NCC* to develop more beneficial information. That is to say, the results of these enhanced operations can be used to (i) verify the origins of *BGP Update* messages by checking the prefix ownership in *RIPE NCC*, (ii) confirm each *BGP* session in the *Path Attribute*s of *BGP Update* messages, (iii) inspect if an *AS* path follows the *Gao-Rexford model* (*GR model*) [17] (i.e., business relationship), and so forth. There is no hard constraint in the sequential order of operations, and each instruction depends on "associated data" in a *BDC* client rather than a topic of analysis. By that means, *BDC* is able to introduce more flexibility in data quality and analysis capability, relatively than other implementations.

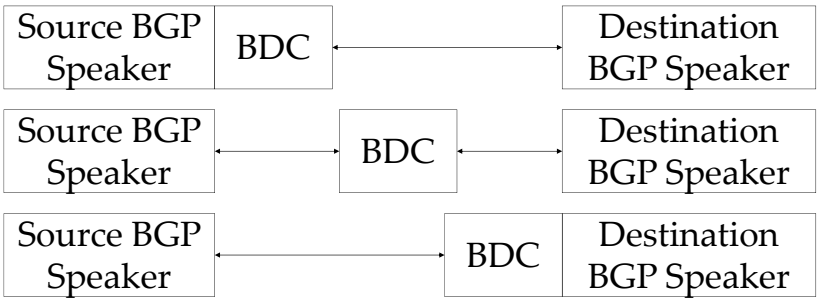


Figure 5-4: The Visualization of a *BGP* Network Architecture Accepting *BDC*

The *Auxiliary Module* in *BDC*

More versatile functionalities are also introduced by the *BDC* through the *Auxiliary Module* that primarily holds the *Packet Organizer* and *Origin Validator*. Figure 5-4 demonstrates that *BDC* is able to be a post-processor, mediator, or pre-processor to supervise the data consistency in the network. As a post-processor, the *BDC* prevents a *BGP* speaker from announcing deceitful *BGP Update* messages. Dropping off malicious packets or inspecting a network in the middle is also possible by the *BDC* for light and reliable traffic. The module does not allow the injection of spiteful data to the *RIB* of a *BGP* speaker when it acts as a pre-processor. *Packet Organizer* introduces the *IP* packet-manipulating functions of the *Network Layer* and *Transport Layer* in the *OSI* model in order to support the portability of *BDC*. That is, the *Packet Organizer* module acts an essential role for the adaptability by organizing and dispersing *BGP Update* messages to *BGP* peers. All fields in a *BGP* packet created by *Packet Organizer* follows the standards [1, 174] to directly handle *IP*, *TCP*, and *BGP* header, and the body of *BGP*; for example, the *IP Prefix* 223.232.0.0/16 is represented as 10 df e8 in *BDC* as in a *BGP* packet. However, in order to successfully disseminating packets, a *BDC* client needs to satisfy an additional requirement that is to know a *TCP* sequence number for an associated *BGP* session because the current version of *BDC* does not handshake with *BGP* speakers. In other words, *BDC* currently does not follow the Finite State Machine (FSM) of *BGP*. By that means, the sniffer module of

a *BDC* server is developed and tracks down the latest *TCP* sequence number on a local *TCP* port in order to send out a packet with a legitimate sequence number. The removal of this functionality will require *BDC* to import the FSM for the same operation. As a consequence, the independence between our implementation and *BGP* speakers does not require any modification of the original *BGP* structure to adopt *BDC* in *BGP* networks.

Incidentally, the *Origin Validator* module in a *BDC* client utilizes the *RIPE NCC RPKI Validator* [175] to validate origins, due to the high level of structural completeness presented by *RPKI* rather than introducing a novel method. A *BDC* client accepts multiple cash servers for reliability to tolerate the connection failure to a server; at this moment, three cash servers are associated with the *Origin Validation* module. Among the servers, a local cache server (localhost:8080) will be preferred to the server deployed by *LACNIC* (*AS*28000) to the one from Kaia Global (*AS*251). The real-world *ROA*s to validate origins are extracted through the following nine *Trust Anchor Locator*s (*TAL*s) [176]: *AfriNIC RPKI Root*, *APNIC from AfriNIC RPKI Root*, *APNIC from ARIN RPKI Root*, *APNIC from IANA RPKI Root*, *APNIC from LACNIC RPKI Root*, *APNIC from RIPE RPKI Root*, *ARIN*, *LACNIC RPKI Root*, and *RIPE NCC RPKI Root*.

Chapter 6

BGP Data Analysis: Load Reduction on Validating *BGP* Paths Based on *BGP* Data Analysis

We here propose a *Validating Load Reduction* (*VLR*) that reduces the burden of *BGP* speakers in validating paths, latency, as well as throughput. The *VLR* disproves the paths that do not conform to the *Gao-Rexford model* (*GR model*) by discovering tiers of *Autonomous Systems* (*AS*es) without any change of infrastructure. In other words, the *BGP Update* messages that bring out route leakage, hijacking, etc., mostly will not be accepted into the *Routing Information Base* (*RIB*) of a *BGP* speaker. Furthermore, *VLR* can locate the malicious paths already stored in *RIB* by utilizing a list of stable paths. The feasibility, deployability, and hands-on experience of *VLR* is also presented by simulations using the *BDC* discussed in Section 5.2.

6.1 The Problem Definition

None of the implementations discussed in Section 5.1 literally analyzes *BGP* data. The volume of measurement data is huge and the extraction of important information is challenging. The *REST API* supported by *RIPE NCC* may be mostly close to the *BGP* data-analyzing tool we go for, due to its interactive

operation. However, likewise to the other tools, it also provides just statistical or historical *BGP* data rather analyzes those data. Therefore, we decided to develop a comprehensive platform that is capable of really analyzing *BGP* data as a standalone system. As a result, we implemented *CLI*-based BGP data-analyzing tool, which is *BDC* introduced in Section 5.2.

In *Path Validation* of *BGP* networks, we concentrate on which updates have malicious *AS* paths and which paths are considered invalid in *Loc-RIB*. The delay to process a *BGPsec Update* message is in the range of 0 to 570 milliseconds, depending on the number of *AS*es in the *Path Attribute* and the number of *NLRI* in the message as discussed in Chapter 4. In other words, it is assumed in the same simulation environment that around one million seconds would be required to verify all *AS* paths in *Loc-RIB* that introduces 2, 872, 386 unique *AS* paths and 3.80 of the *AS* path length in average; these numbers are extracted from the report of the *RouteViews* project [122]. Rexford et al. [177] found that most prefixes tend to have stable *BGP* routes for days or weeks. In addition, the *BGP* does not require a periodic refresh of the routing table [1]. In other words, if any malicious path has been already accepted into the *Loc-RIB* of a *BGP* speaker, then it is likely to stay and be facilitated to cause similar *BGP* accidents discussed in Section 2.2. Although *AS* paths in *BGP Update*s are more likely to be validated rather than those in *RIB* entries, and the total number of paths in *BGP Update*s is generally much smaller than that in *RIB* entries in the same time frame, zero to few hundred milliseconds in delay for each path will be definitely considerable to *BGP* speakers in terms of that a regular *BGP Update* message takes 0.4 millisecond to be processed.

By that means, we developed *VLR* which statistically analyze *BGP* data in *BDC* and reduces the load to validate *BGP* paths. *BGPsec* is expected to be pessimistic on validating *BGPsec Update* messages in a "reasonable" time. To the best of our knowledge, concluding valid paths in a moderate period is difficult, and the *BGPsec* algorithm will be the easiest and secure way for it.

However, what if we can reduce the number of updates to be validated? We toed the line on this idea and focused on identifying "invalid" paths instead of valid ones.

6.2 The Implementation of *Validating Load Reduction* (*VLR*) in *BDC*

6.2.1 Designing the Base of the *Gao-Rexford model* (*GR model*)

The first requirement of a valid path in *BGP* networks is to comply with the *GR model* [17]. It states that a valid path should be one of following six types in shape of tiers [16]: (i) an uphill path, (ii) a downhill path, (iii) an uphill path followed by a downhill path, (iv) an uphill path followed by a peer-to-peer edge, (v) a peer-to-peer edge followed by a downhill path, (vi) or an uphill path followed by a peer-to-peer edge that is followed by a downhill path. Each of these six paths is referred to as a *GR Path* in this study. In other words, these six types can be used as criteria to distinguish invalid paths from valid ones. However, the *Path Attribute* of a *BGP Update* message contains only *ASN*s in a sequence. It is not enough to get notified about the levels (i.e., tiers) of *ASN*s in a path to differentiate sessions in path type. Figure 6-1 is captured in *BGPlay* and visualizes the portion of the real-world *BGP* network centered by *AS*174 at a certain time. Speaking of which, a higher tier *AS* (e.g., *AS*174 which is one of the *Tier-1* networks) and a lower tier *AS* are in a one-to-many relationship due to the *IP* address-allocating structure; that is, as far as the *IP* allocation is established for the monetary benefit of providers, this business relationship would not be impaired between *AS*es. At the first time, we assumed in inter-*AS* relationship that

Observation 6.2.1. a higher tier *AS* usually has more neighbor counts

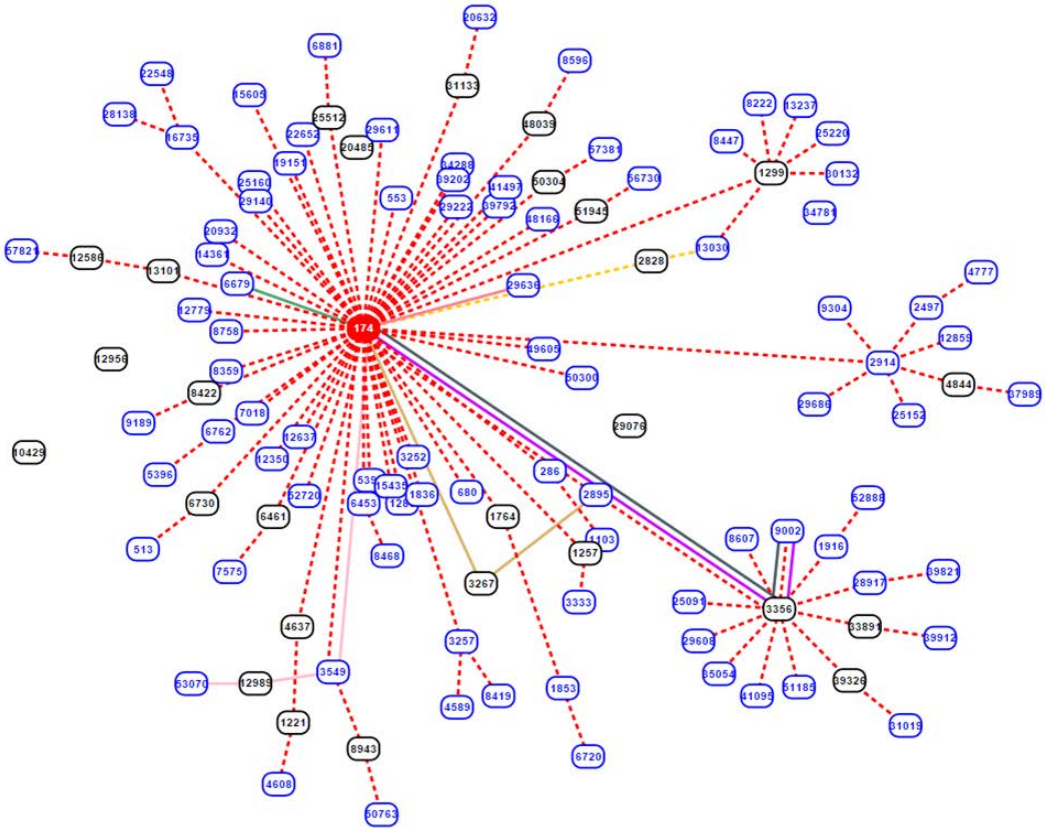


Figure 6-1: The *BGP* Network Observed in *BGPlay* on the Perspective of *AS*174

as shown in Figure 6-1 in such a way that tiers of *AS*es in *AS* paths can be denoted. Note that researchers [178] present a heuristic algorithm that classifies *AS* relationships from *BGP* routing tables. The algorithm categorizes more than 90.5 percent of *AS* pairs into provider-customer relationships, less than 1.5 percent of *AS* pairs into sibling relationships, and less than eight percent of *AS* pairs into peering relationships.

However, the total number of *ASN*s at that moment was 84, 182 [179], except *Bogon*s and *Full-Bogon*s. That is to say, corresponding *BGP*-data crawling is not easy. Speaking of which, exchanging an *HTTP* request and response with the *RIPEstat* database takes around two seconds resulting in significant delay; that is, all 84, 182 *AS*es' neighboring information will be gathered approximately in 190, 000 seconds (or 2.2 days). In order to mitigate this exhaustion in time, we have developed a two-dimensionally threaded

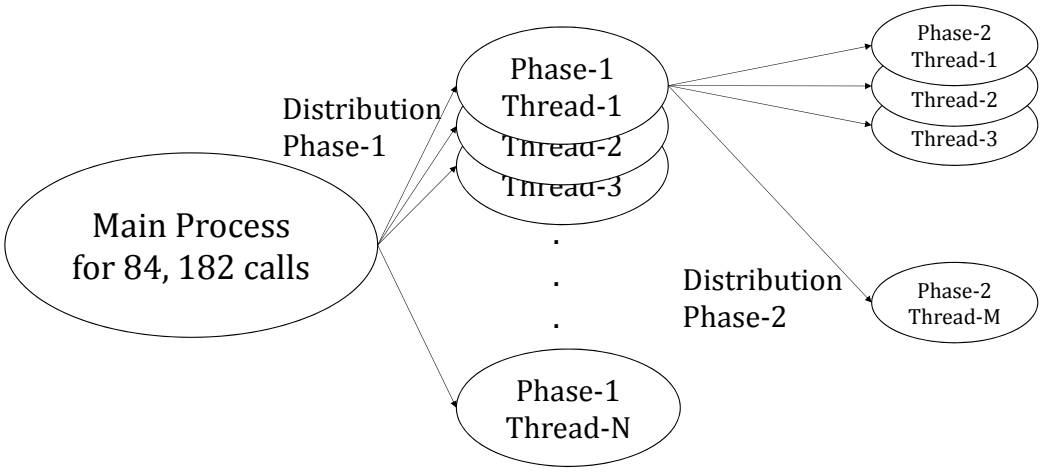


Figure 6-2: The Design of Two-dimensionally Threaded *BGP* Data Crawler

crawler as depicted in Figure **??**, and this crawler is embedded into the *Information Builder* in Figure 6-4, which represents the structural overview of *VLR*. In Figure 6-2, each *Phase-2* thread generates a single *HTTP* request for the neighboring information of a single *ASN* and records the corresponding response. The number of *Phase-2* threads, M, is determined by the number of *Phase-1* threads, N. Our simulation on 3.1-Ghz dual-core *CPU* reveals that our proposed structure to crawl *BGP* data from *RIPEstat* database is able to fulfill all 84, 182 *HTTP* requests within four hours; that is, the delay is reduced by 87 percent. This simulation result is derived when $M = 500$ and $N = 169$; note that M and N have not been "optimized" because of the satisfactory performance.

The neighboring information of *ASN*s can be collected from the *RIPEstat* database, and the *RIPEstat DATA API* provides two *HTTP* calls for the information, which are *ASN Neighbors* and *ASN Neighbors History*. We have leveraged both types to gather all available neighboring information and found no remarkable difference between their data, except that *ASN Neighbors* additionally offers geographical information. By that means, *ASN Neighbors* is selected for *BGP*-neighboring data collection; in addition, 383, 302 sessions are found by *ASN Neighbors* while 337, 141 are detected by *ASN Neighbors History*. The neighboring information extracted through the *ASN Neighbors*

request is referred to as *neis*. The *neis* is structured in *JSON* and has the neighboring information of the 84, 182 *AS*es (i.e., *AS*1 to *AS*395, 164 without *Bogon*s and *Full-Bogon*s) available in the *RIPEstat* database. The information for each *ASN* contains zero or more {*ASN*, geographical position} pairs. Figure 6-3 demonstrates the neighboring or prefix-owning information retrieved from the three different types of requests as examples. Only speaking of *neis*, the neighboring information of an *ASN* and the geographical location of those neighbors will be obtainable. For instance, *AS*50088 has a *BGP* session to *AS*174, *AS*45531, *AS*6930, *AS*8220, *AS*43610, and *AS*13237. Four of all six sessions are geographically located on the west side. When the neighboring information of an *ASN* does not exist in the *RIPEstat* database, *neis* will introduce a truncated array for the *ASN*, likewise to *AS*60141 in the figure. At the time of crawling data, no data was returned from the database for 32, 086 *ASN*s. In other words, it is assumed that 38.12 percent of total *ASN*s may not have neighbors or may not provide their information to *RIPE NCC*. Gregori et al. [180] found by analyzing *BGP* data gathered by *RouteViews*, *Routing Information Protocol* (*RIP*), and *Packet Clearing House* (*PCH*) that large areas of the *Internet* are not properly captured due to the geographic placement of the current route collector feeders and due to *BGP* filters, such as *BGP* export policies and *BGP* decision processes. It is because only large *AS*es typically contribute to the collectors. The incompleteness of capturing can be efficiently mitigated either by increasing the number of *BGP* feeders that are multi-homed or by following the *do ut des* principle. *BGP* sessions could be also checked by active *Internet Control Message Protocol* (*ICMP*) ping and *traceroute* measurements. However, researchers argue the usage of *traceroute* on inspecting network topology as discussed in Section 2.3.3. Accordingly, an *AS* path is considered invalid in evaluating *GR model* if any of *BGP* session in the path is not evidenced by *neis*.

```
"50088" [[174, u'left'], [43531, u'left'], [6939, u'left'], [8220,
 u'left'], [43610, u'right'], [13237, u'uncertain']]
60141 []
```

(a) An Example of *neis* in Structure


```
"50088",
{
    "195.211.224.0/22": {
        "owned": {
            "lower_bound": 3285442560,
            "upper_bound": 3285443583
        }
    },
    "2001:978:5200::/48": {
        "owned": {
            "lower_bound": 42540680236419548142654978811267907584,
            "upper_bound": 42540680236420757068474593440442613759
        }
    }
}
```


(b) An Example of *hier* in Structure

```
"50088" [174, 6939, 8220, 43531, 43610]
```

(c) An Example of *neih* in Structure

Figure 6-3: The Example of *neis*, *hier*, and *neih* Based on the *RIPEstat* Database

Acquiring and Manipulating *BGP* Data Provided By *RouteViews*

We utilize our implementation *BDC* discussed in Chapter 5.2 to crawl *BGP* data. In detail, it gathers routing information from *BGPmon*, such as the *RIB* entries or *BGP Update* messages. This implementation is also able to download archives on the *RouteViews* website. The subjected archives are *table dumps*, of which each provides the *RIB* entries dumped every two hours, and *update dumps* that stores the *BGP Update* messages coming in a 15-minute interval. Each type is usually sized in 78 and one megabyte around so, and *RouteViews* creates 12 *table dumps* and 96 *update dumps* for each *BGP* speaker on a daily basis, resulting in approximately 936 and 96 megabyte-sized data in a day. Note that *BDC* is also featured in selectively downloading those archives by name or period; for instance, a selective download for a month will transfer 360 archives for *RIB* entries and 2880 archives for updates with a single command.

The files downloaded from *RouteViews* are in *MRT* format so they are reprocessed and reformatted into a *BDC*-specialized data structure called *as-data* that consists of four *Counter*-typed variables which are *All AS Paths*, *All ASNs*, *stub-ases*, and the *remnant* in *JSON* format. Each of the variables is very intuitively named. In specific, the *All AS Paths* variable owns all *AS* paths allowing duplicates, and the *ALL ASNs* variable possesses all *ASN*s in those paths; duplicates are also permitted. If an *AS* has a single neighbor in *neis*, it is considered an *stub-ases* entry. The *remnant* consists of *AS*es that have no neighbor found in *neis*. The *as-data* of the *update dump* created by *RouteViews* on September 2, 2015 at midnight consists of 21, 865 unique paths, 1588 unique *AS*es, 243 unique *stub-ases*, and five unique *remnant*s. Note that all archives are retrieved from the *BGP* speakers of *route-views2.oregon-ix.net* under the *RouteViews* project.

Accumulating and Handling BGP Data Offered By RIPEstat

BDC is also leveraged to collect data from the *RIPEstat* database. Three types of *RIPEstat* data calls are utilized to gain the neighboring and prefix-owning information of *ASN*s; that is, *ASN Neighbors*, *ASN Neighbors History*, and *Routing History*. The two-dimensionally threaded crawler visualized in Figure 6-2 is facilitated for all three different data types to significantly lessen the network delay. The data structure that is specialized in *VLR* and stores all historical prefix-owning information from the *RIPEstat* database is called *hier*. In *hier*, the upper and lower bound of the owned prefixes are accessible in numeric interpretation as captured in 6-3b. The *neih* structure is constructed on the basis of the neighboring information collected through the *ASN Neighbors History* call, which is a similar call to *ASN neighbors*, and the detailed structure of *neih* is illustrated in 6-3c. As discussed in Section 6.2.1, the difference between *neis* and *neih* are inconsiderable, and *neis* is chosen for all simulations. Note that *neis*, *hier*, and *neih* are respectively about 8.1, 164.8, and 3.3 megabytes in size.

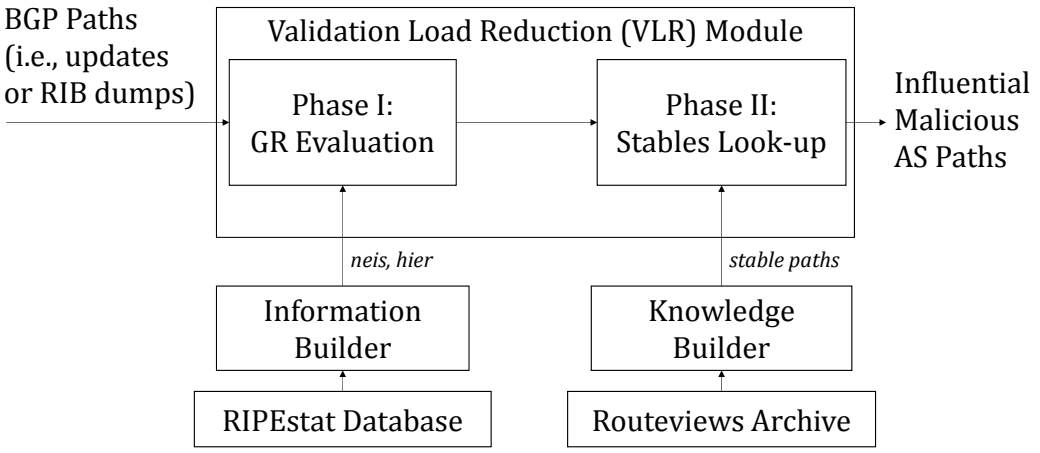


Figure 6-4: The Structural Diagram of the *Validating Load Reduction* (*VLR*) Module

6.2.2 The Details of Implementing *VLR*

Basically, a *BGP* speaker requires validating the *Path Attribute*s of all incoming *BGP Update* messages in order to criminalize the fabricated *AS* paths. An *update dump* in a 15-minute interval from *RouteViews* has about 60, 000 messages. It means that a related *BGP* speaker needs to take care of 67 update messages in a second; in other words, one *BGP Update* message is supposed to be validated and processed in 15 milliseconds by the speaker. However, processing a *BGP Update* message will take much longer with *BGPsec* as discussed in Chapter 4. Therefore, we have implemented the *Validating Load Reduction* (*VLR*) model as depicted in Figure 6-4. The VLR module primarily consists of two phases: the *GR Evaluation* and *Stables Look-up* phase.

First Phase—Non-*GR Path*? No Trespassing!

The assumption that all of the valid *BGP* paths should follow the *GR model* can be easily proved in a deduction manner as follows. *BGP* sessions are established on the basis of the *IP* allocation that introduces a pecuniary relationship between *AS*es. Due to the relationship, these paths are preferred

for *BGP* traffic to the other paths that do not include a monetary relationship. In other words, a provider should propagate *IP* traffic to/from its customer in an obligatory fashion. Plus, this obligation is unidirectional and given to providers. In other words, once traffic is received from a peer or provider, the traffic cannot be advanced to another peer or provider. These sessions (i.e., from/to a customer *AS*) will be indicated by a higher *Local Preference* value in real-world routing so the paths will be naturally prioritized because of the *Best Path Selection Algorithm* (*BPSA*) in *BGP* [15]. In conclusion, the final shape of an *AS* path in tier should be one of six *GR Paths* introduced in 6.2.1. Therefore, we establish the observation,

Observation 6.2.2. All valid *AS* paths in *BGP* act in accordance with the *Gao-Rexford model* (*GR model*).

The *neis* is leveraged as base data to examine Observation 6.2.2, based on which all *AS* paths are categorized into *invalid* or *not invalid* consecutively through the *peak selection* and *path validation in shape* in Algorithm 1. The algorithm, which elaborates the construction base of our path-weighting algorithm in a Python-friendly format, examines all paths so as to differentiate non-*GR Paths* by following the two steps:

(i) *peak selection*: At this stage, the highest tier *AS* (i.e., *peak*) in a path is determined, and the expected number of *peaks* is one or two. In other cases (e.g., three or more *peaks*), the path under inspection will be considered invalid.

(ii) *path validation in shape*: Corresponding to the number of *peaks* in an *AS* path, the path is evaluated if it is in a triangle or trapezoid shape. These two shapes originate from the six path types discussed in 6.2.1.

Second Phase—Spotting Criminals in the Heaven

Again, the basic idea of *VLR* is to detect "invalid" candidates in paths. However, if a *BGP* speaker somehow does not accept those candidates into

Algorithm 1 The Construction Base of Evaluating a *GR Path*

```
gao_rexford = False
apw = [len(neis[e]) for e in AS_Path]
peak = []
if not peak then
    tmp_tuple = [(e, len(neis[e])) for e in AS_Path]
    peak = [max(tmp_tuple, key = lambda x:x[1])[0]]
end if
if len(peak) == 1 then
    peak = peak[0]
    if apw[:AS_Path.index(peak)+1]==sorted(apw[:AS_Path.index(peak)+1])
    and apw[AS_Path.index(peak):]==sorted(apw[AS_Path.index(peak):],
    reverse=True) then
        gao_rexford = True
    else
        gao_rexford = False
    end if
else if len(peak) == 2 then
    if apw[:AS_Path.index(peak[0])+1]==sorted(apw[:AS_Path.index(peak[0])+1])
    and apw[AS_Path.index(peak[1]):]==sorted(apw[AS_Path.index(peak[1]):],
    reverse=True) then
        gao_rexford = True
    else
        gao_rexford = False
    end if
else
    gao_rexford = False
end if
return gao_rexford
```

its *RIB*, then this filtering will be actually unnecessary. By that means, *VLR* concentrates on "*malicious influential*" paths among invalids in this phase. The *malicious influential* path is an *AS* path that has survived in the *RIB* (i.e., *Loc-RIB*) of a *BGP* speaker and will very likely produce despiteful results in the network. In order to accomplish this task, *VLR* requires a list of influential paths that will be referred to as *stable* paths or static paths. The *stable* (a.k.a. influential or static) path means an *AS* path that has remained in *Loc-RIB* for a certain amount of time; in our simulations, the duration to decide the stability of an *AS* path is 24 hours. The decision comes from that most

past accidents [44, 46–49, 52, 53, 55, 181] were acknowledged and corrected by network administrators in hours. The *Knowledge Builder* in Figure 6-4 initiates *stable* paths based on the *RIB dumps* of *RouteViews*. The module retrieves all unique *AS* paths from an *RIB dump* and locates the intersection with another *RIB dump*. It iterates the responsibility until all dumps in a given period are evaluated. At the last iteration, the intersection will be considered *stable* paths. Due to the duration we set up for *stables*, twelve files and eleven repetitions are subjected to the production of *stables*. At the end of this phase, the *stable* paths will be compared to the deceitful paths anticipated in the *GR Evaluation* so that the *malicious influential* paths can be enumerated.

In summary of 6.2.2, *VLR* is featured in separating out invalid candidates (i.e., non-*GR Paths*) and evaluate if there is any possibility that those candidates are deposited into the *RIB* of a *BGP* speaker by any chance. In order to examine *VLR*, we have installed three simulations. Those simulations are differentiated in the incoming *AS* paths. The first simulation is designed to acquire virtually created *AS* paths. These are neither from *RIB* or *BGP Update*s. This experiment is just concentrating on evaluating *GR Paths*. It is forecasted in this simulation that most virtual paths do not comply with the *GR model*. In the second simulation, real-world *BGP Update* messages are going to be investigated by *VLR*. Namely, the "practical" proportion of non-*GR Paths* and *malicious influential* non-*GR Paths* are disclosed. The last simulation is organized to see if it can find out the maliciously influential paths that precipitated one of past real-world accidents. The selected accident for this simulation is the Google's Outage occurred in 2012 [53] because this is the latest and well-known accident at that moment. Note that all simulations in Section 6.3 are performed on 64-bit Ubuntu 14.04 LTS that runs on 3.1-GHz dual-core *CPU* with four gigabytes of memory.

Table 6.1: The Number of Invalid/Survived *AS* Paths in the Randomly Generated *AS* Paths

	AS Path Counts	
	Total (in Percent)	Unique (in Percent)
Random AS Paths	3, 030, 000 (100%)	2, 986, 197 (100%)
Invalid AS Paths	2, 975, 907 (98.2%)	2, 941, 342 (98.5%)
Survived AS Paths	54, 093 (1.8%)	44, 855(1.5%)

6.3 The Simulation Results of *VLR*

6.3.1 The Validation of Randomly-Generated Paths

3, 030, 000 *AS* paths have been randomly produced to check if *VLR* is able to filter out invalid paths. Each *AS* path consists of *AS*es in the range of one to 395, 164 in number and has one to fifteen in path length. Table 6.1 demonstrates the validation results of those *AS* paths produced by *VLR* in the count. After removing duplicates, the total number of unique *AS* paths were 2, 986, 197, among which 98.5 percent is considered invalid and the others, 1.5 percent, is survived from the *GR Evaluation* in *VLR*. Speaking of the 44, 855 paths, these are not non-*GR Paths*; in other words, they may be valid or invalid paths. That is, if a *BGP* speaker embeds the *VLR* module and receives the three million *BGP Update* messages having the same path information, it will be required by *VLR* to evaluate only 44, 855 paths for 54, 093 *BGP Update* messages. This experiment reveals that the *Information Builder* can very significantly decrease the validation load of a *BGP* speaker for randomly generated paths maliciously aiming at routing overload.

6.3.2 The Validation of Real-world *BGP Update*s

In a more practical manner, *VLR* has been utilized to review the *AS* paths in the *update dump* that stores the information of real-world *BGP Update*

Table 6.2: The Simulation Results of the Experiment Specified in 6.3.2

(a) *GR Evaluation* Analysis

Total number of *BGP* paths in the *BGP Updates*	
21, 865 (100%)	
True Positives+False Positives	*True Negatives*
19, 331 (88.41%)	2534 (11.59%)

(b) *Stables Look-up* with the *Stables* on September 1, 2015

Stable Paths on September 1, 2015	Updates ∩ *Stables*	
2, 935, 122 (100%)	7, 447 (100%)	
Stable Paths on September 2, 2015	*Positives* ∩ *stable* paths	*Negatives* ∩ *stable* paths
2, 908, 522 (100%)	7132 (95.77%)	315 (4.23%)

(c) *Stables Look-up* with the *Stables* on September 2, 2015

Total number of paths		z = (*N*∩s on September 1)
7, 116 (100%)		∩
Positives ∩ *stables*	*Negatives* ∩ *stables*	z∩s on September 2
6823 (95.88%)	293 (4.12%)	239

messages received by the *BGP* speakers of *route-views2.oregon-ix.net* between 11:45pm and midnight on September 2, 2015. To the best of our knowledge, there were no issues in the *BGP* networks associated with the associated *BGP* speakers at that time, so it is anticipated that most paths will be stable and valid. Table 6.2 informs the results of this experiment. The *GR Evaluation* in *VLR* first inspects 21, 865 *AS* paths in the update archive. The number of survived paths in this step is 19, 331; that is, 88.41 percent of the paths requires to be additionally scrutinized by a *BGP* security solution such as *BGPsec*. On the other hand, 11.59 percent can be excluded from further validations so it states that this network is very stabilized. For the *Stables Look-up*, we first downloaded all twelve *RIB* archives possessing the *RIB* entries on September 1, 2015 to develop a list of *stable* paths. Figure 6-5 visualizes the iterations to create the list. The x-axis represents the time line in a two-hour interval on September 2, 2015 so each *RIB dump* file in a corresponding period can be

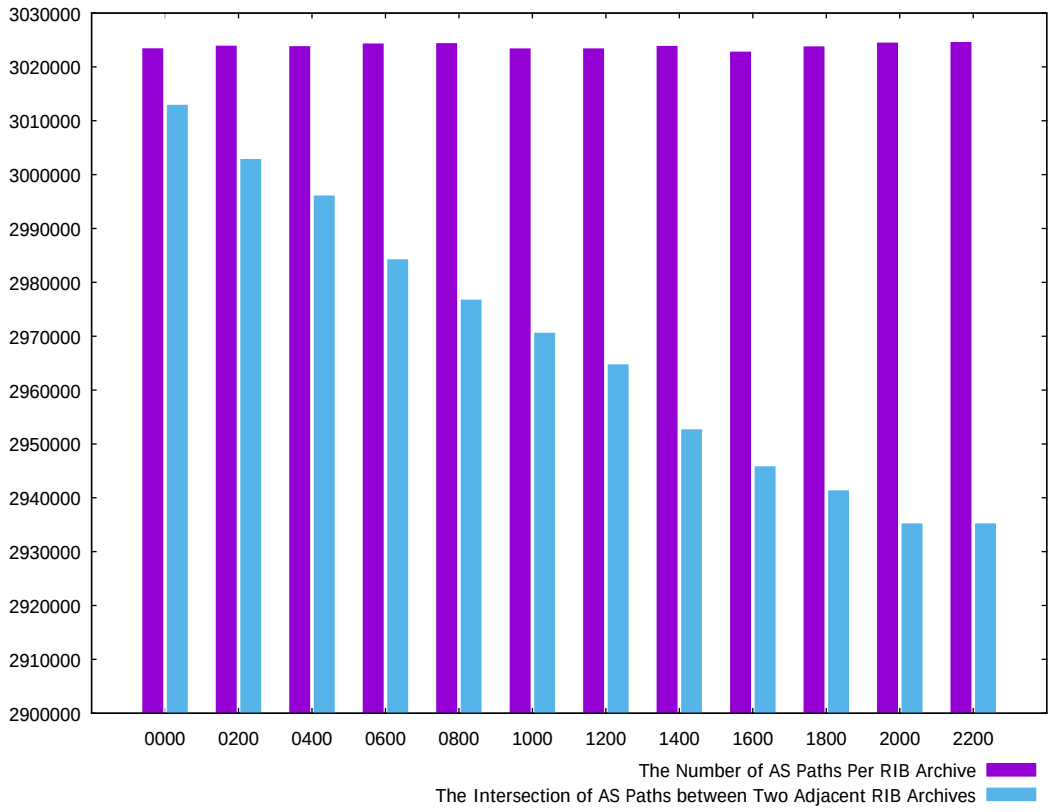


Figure 6-5: The Extraction of *Stable AS* Paths from the *RIB* Archives Provided by *RouteViews* on September 1, 2015

considered an x-tick on the graph. In average, the number of total *AS* paths in an archive is 3, 023, 760. As the comparison of two adjacent archives is repeated, the number of *AS* paths in the intersection is gradually decreased, resulting in 2, 935, 122 stable paths at the end; that is, 97.07 percent of the *RIB* entries of *AS*6447 can be considered *stable*. It is noted from the analysis that a *BGP* speaker barely updates its *RIB* although it may include potential threats. Table 6.2b demonstrates the consequences of the *Stables Look-up* for this simulation. Note that the outcome of the *GR Evaluation* is *invalid* or *not invalid*; those two types are referred to respectively as *True Negatives* and *Positives* in Table 6.2a. On the optimistic analysis of the result in 6.2b, 36.89 percent of all positives (i.e., $\frac{7132}{19{,}331} \times 100$) are considered *stable* so a *BGP* speaker, in this case, can possibly validate the rest, which is 55.79 percent (i.e., $19,\ 331 - 7132 = 12,\ 199$) by dismissing the *stable* paths already in its *RIB*.

In other words, *VLR* is able to decrease the validation load approximately by 44 percent (i.e., $100 - \frac{7132+2534}{21,\ 865} \times 100$) on positive analysis. On the pessimistic analysis, *VLR* easily enumerates spiteful paths, which occupy 11.59 percent, without actual validation. *VLR* is able to observe the influence of those paths and points out 315 paths among them through the *Stables Look-up*. These 315 paths are suspected of *malicious influential* troublemakers. In order to analyze the remained/withdrawn ratio of stable and *malicious influential* paths, the list of *stable* paths on September 2, 2015 is additionally retrieved; from the generation similar to Figure 6-5, the list introduces 2, 908, 522 *stables*. Our analysis reveals that whereas $AS6447$ changes approximately five percent of its *RIB* entries in a 24-hour period, it withdraws 24 percent of *AS* paths in the list of *malicious influential* paths in the same period (i.e., 76 out of 315). In other words, we can say that these were also recognized by $AS6447$ as unstable at least or malicious at most at that moment.

6.3.3 Case Study: Google's Outage 2012

As another practical simulation, we have analyzed the *BGP Update*s and *RIB* entries when the Google's Outage took place in 2012. The accident occurred at around 02:24 *UTC* on November 6, 2012. The *Cloudflare* reports that they saw the malicious *BGP* routes for a Google *IP* address ($AS15169$), which traversed Indonesian *ISP* called *Moratel* ($AS23947$). One of the *AS* paths that they found from their place ($AS13335$) is $AS4436$, $AS3491$, $AS23947$, followed by $AS15169$. The $AS13335$ is geographically not far from $AS15169$ so packets should never been routed through $AS23947$. So as to know if the accident affected the routing table of $AS6447$ at that time, we analyzed the related archives of *BGP Update*s and *RIB* entries. We found in the analysis that the update archive dumped on November 6 2012 at 02:30am includes the following malicious paths: '$AS13030$ $AS3491$ $AS23947$ $AS15169$ $AS6432$,' '$AS13030$ $AS3491$ $AS23947$ $AS15169$,' '$AS13030$ $AS3491$ $AS23947$ $AS15169$ $AS19425$,' and '$AS13030$ $AS3491$ $AS23947$ $AS15169$ $AS26910$.' So

we installed a simulation to see if those paths can be detected by *VLR*.

For the scrutiny of the accident, the 24-hour *stable* paths on November 5, 2012 have been produced in the *Stables Look-up* in *VLR*. Table 6.3 confirms the statistical results of our simulation regarding the accident. Among the 15, 870 number of *BGP Update* messages, the *GR Evaluation* of *VLR* differentiates 2327 malicious paths occupying 14.66 percent, in which all of the four invalid paths above was observed. Moreover, *VLR* discovers 715 spiteful paths in the *RIB* as shown in Table 6.3b. In contrast to that the proportion of the influential candidates in *BGP Update*s is just 1.44 percent in the simulation provided in Section 6.3.2, the proportion in this simulation is 4.51 percent. In other words, $AS6447$ was propagating much more malicious paths than common days at the time of the accident. Additionally, 56 out of the 715 paths include $AS23947$ so we insist that many more malicious paths do exist in the list. This assertion also can be supported by the number of *BGP* paths having $AS23947$ at the time of the accident as demonstrated in Table 6.3c. Speaking of which, the duration of hijacking accident was 30 minutes from 02:24am around on November 5, 2012, and the sharp increment of path counts at 03:00am is likely to originate either from the fluctuation of paths in the *RIB* or from the detours selected by *BGP* speakers to abstain $AS23947$ in paths.

6.4 The Enhancements and Discussions of *VLR*

This section introduces a few considerations that may weaken our contribution and the countermeasures approaching those problems.

6.4.1 The *Flex Path-generating Algorithm* (*FPA*) for *GR Paths*

The first possible problem of *VLR* is that Algorithm 1 possibly produces *False Negatives*, which are *GR Paths* that are wrongly considered invalid in the *GR*

Table 6.3: The Simulation Results of the Experiment Detailed in 6.3.3

(a) *GR Evaluation*

Total number of *BGP* paths in the *BGP Update*s	
15, 870 (100%)	
True Positives+False Positives	*True Negatives*
13, 453 (85.34%)	2, 327 (14.66%)

(b) *Stables Look-up* with the *Stables* Derived on November 5, 2012

Stable Path Counts on November 5, 2012	Updates ∩ Stables	
1, 705, 461	7, 045 (100%)	
	Positives ∩ *stable* paths	*Negatives* ∩ *stable* paths
	6, 330 (89.85%)	715 (10.15%)

(c) The Number of *BGP Update*s on November 5, 2012 Dumped at

	02:15am	02:30am	02:45am	03:00am
Total # of *BGP* Paths	14, 054	15, 870	15, 828	21, 724
BGP Paths including *AS*23947	531	3777	546	233

(d) The Number of *AS* Paths on November 5, 2012 Dumped at

	02:15am	02:30am	02:45am	03:00am
AS Paths including *AS*23947	1444	2321	1639	1174

Evaluation of *VLR*. For example, an *AS* path (53364 3257 3356 35805 58185 58185 58185) will be falsified by the algorithm. It is because the weights of *ASN*s in the path are 2, 2768, 4337, 73, 2, 2, and 2 in *neis*. It means that Algorithm 1 is incompatible with the *AS-Path Prepending* of *BGP*; in specific, the triple of *AS*2 at the end is recognized by *VLR* as two peer-to-peer sessions. This problem is solved by adding a preprocessor of the *GR Evaluation* to eliminate the *AS-Path Prepending*.

Algorithm 1 possibly develops another problem due to its strict validation. Speaking of which, each *ASN* in a path is weighted corresponding to the number of neighbors attained from *neis*. Intuitively, an *AS* having a bigger number than the other in the count will be considered a higher tier. Of course,

Table 6.4: The Number of Neighbors (Weights) of Top Tiers by Occurrence

Rank	ASN	# of Neig.	# of Occurr.	Rank	ASN	# of Neig.	# of Occurr.	Rank	ASN	# of Neig.	# of Occurr.
1	3356	4337	530, 028	11	209	1680	60, 591	21	20485	2017	17, 950
2	174	4864	321, 371	12	13030	2019	55, 838	22	4323	1870	17, 658
3	6939	7856	272, 718	13	6762	752	47, 925	23	8359	1818	17, 633
4	2914	2611	245, 095	14	6461	1518	44, 901	24	3303	609	17, 383
5	3257	2768	235, 240	15	2828	1173	34, 313	25	1267	810	15, 896
6	1299	1198	170, 541	16	701	1507	28, 818	26	31133	1473	14, 582
7	7018	4656	132, 417	17	1239	631	25, 022	27	5580	936	13, 041
8	6453	1308	100, 064	18	286	898	20, 713	28	19151	1346	11, 003
9	3549	2111	69, 606	19	3491	610	19, 397	29	37100	192	10, 937
10	9002	3799	67, 971	20	3267	828	19, 202	30	2497	553	10, 751

this cannot be an infrangible criterion. For instance, an *AS* path (3561 4637 4637 4637 4637 9901 9901 9901 4768 18021) respectively has the weights of 286, 283, 283, 283, 283, 28, 28, 28, 37, and 2. After removing the *AS-Path Prepending*, the weights will be 286, 283, 28, 37, and 2. This path possibly turns out to be *False Negatives*. We found in real-world *BGP* data that determining tiers based on the number of neighbors is getting blurred at a lower tier. Table 6.4 also concurs in permitting the tolerant validation. It exposes the top 30 *ASN*s by occurrence among 919 peaks retrieved from the 2, 935, 122 unique and *stable* paths received by *AS*6447 on September 1, 2015. The *ASN*s listed are likely to be *Tier-1* speakers (or probably *Tier-2*) due to the number of occurrences. Note that ***Tier-1*** network is a transit-free network, ***Tier-2*** network peers with some networks and purchases IP transit, and ***Tier-3*** network needs to purchase transit from other networks to participate in the *Internet*. The top 30 *ASN*s have the diverse number of neighbors in the range of 192 to 7856. Furthermore, *AS*8492, *AS*293, *AS*2152, and *AS*5413, which are in the top 100 *ASN* list by occurrence, respectively have the weight of 61, 57, 83, and 99 in *neis*. In summary, we conclude from the data analysis that categorizing *ASN*s in the tier is not available in an absolute manner and should be determined in a relative fashion. Accordingly, we have devised the flexible GR-evaluating algorithm as in Algorithm 2, which is referred to as the *Flex Path-generating Algorithm* (*FPA*). It categorizes *ASN*s in a tier by

"negotiably" comparing two adjacent *AS*es. In specific, at the erroneous step of 1, this algorithm will consider two contiguous *AS*es of similar weights *en route* such as the *Peer-to-Peer* relationship in *GR model*. Note that the *FPA* is, however, not employed in our simulations because this algorithm probably produces *False Negatives*.

Algorithm 2 Variation for a Flexible *GR Path*

```
1: flex= []
2: prev= []
3: for all AS_Path do
4:    if prev then
5:       if len(flex)-1<AS_Path.index(max(AS_Path)) then
6:          if prev>each && (math.floor(each/prev) + math.floor(prev/each)) == 1 then
7:             flex.append(prev)
8:          else
9:             flex.append(each)
10:         end if
11:      else
12:         if prev<each && (math.floor(each/prev) + math.floor(prev/each)) == 1 then
13:            flex.append(prev)
14:         else
15:            flex.append(each)
16:         end if
17:      end if
18:   else
19:      flex.append(each)
20:   end if
21:   prev= each
22: end for
```

6.4.2 The Prefix-based Classification of Tiers

The second problem of *VLR* is the demand of entire neighboring information in *neis*. As discussed in Section 6.2.1, the *BGP* data derived from the *RIPEstat* database seems to be incomplete. In this section, we introduce an alternative way of *VLR* to group tiers using *hier*. The best way to determine tiers would make use of the information of prefixes owned by *AS*es. Generally, at least, one prefix owned by an *AS* is a portion of the prefix possessed by

its provider due to the business relationship between *AS*es. In terms of that, Algorithm 3 identifies the type of each session in a path (i.e., peer-to-peer, provider-to-customer, customer-to-provider, or sibling-to-sibling) and then designates tiers. Nevertheless, *hier* includes all prefix information that an *AS* has ever announced. In other words, it possibly aggravates the accuracy of Algorithm 3 by developing a noise in data. It is helpful to crawl data for *hier* in a short and selected period to extenuate the noise. Despite that, selectively collected routing-history data significantly lessens the information attainable in *hier*. In specific, the full version of *hier* is missing the routing information of 19, 101 out of 84, 182 *AS*es; in other words, at least, 22.69 percent of routing information will be inaccessible. Moreover, likewise to Algorithm 2, Algorithm 3 highly depends on the perfection of data (i.e., *hier*) so we, instead, have selected Algorithm 1 and *neis* because of the lower dependency between them, as well as no generation of *False Negatives*.

Algorithm 3 *AS* Tier Categorization Based on *IP Prefix*es

```
1: valids = [], cnt = 0
2: prev_upb = hier[prev_as][x]['owned']['upper_bound']
3: prev_lob = hier[prev_as][x]['owned']['lower_bound']
4: curr_upb = hier[curr_as][x]['owned']['upper_bound']
5: curr_lob = hier[curr_as][x]['owned']['lower_bound']
6: for all AS_Path do
7:    ases_in_path= path.split(' '), path_length= len(ases_in_path)
8:    for zero to range(path_length) do
9:       relation= False
10:      if prev_as== curr_as then
11:         continue
12:      end if
13:      for all x=hier[prev_as].keys() do
14:         for all y=hier[curr_as].keys() do
15:            if prev_upb−curr_upb>= 0 and curr_lob−prev_lob>= 0
               then
16:               relation= True
17:               break
18:            else if curr_upb−prev_upb>= 0 and prev_lob−curr_lob>= 0
               then
19:               relation= True
20:               break
21:            end if
22:         end for
23:         if relation== True then
24:            break
25:         end if
26:      end for
27:      if relation== False then
28:         break
29:      end if
30:   end for
31:   cnt += 1
32: end for
```

Chapter 7

Conclusion

Our purpose of the studies introduced in this paper is to secure a network protocol that does not provide enough security level for self-protection. By that means, we selected *BGP*, which is categorized into one of the most vulnerable network protocols because it leaves too much in security to be desired and lays the groundwork for the *Internet*. One way to secure a network protocol is to conduct an experiment in a corresponding real-world network to investigate vulnerabilities. For the experimental network environment, network simulators such as *ns-2* and *GNS3* support the *BGP* inspection in virtual networks. An alternative way is to modify the protocol to enhance its security; regarding which, so many *BGP* contributions have been proposed in past decades as discussed in Section 3.1. As well in the former method, the network simulation in this methodology is also required for the accurate analysis.

Our research is summarized in Table 7.1, speaking of which, we have chosen *RPKI* and *BGPsec* for the *Origin Validation* and *Path Validation* to secure *BGP*. At the time of our research regarding the *Origin Validation*, Quagga*SRx* was the only *RPKI*-capable *BGP* software router. However, its fully virtual network environment, which does not comply with the *RPKI* standard, was unusable for practical validations and corresponding simulations. By that means, we implemented *practically-usable* and *RPKI-capable BGP* software router, which is *RTR-BIRD* v1.0. The problems of *RTR-BIRD* v1.0 are

Table 7.1: The Summary of Our Research

Validation Types	Selected Solution	Software Router for Experimental Network	Simulation of the Network	Enhancement of the Solution
Origin Validation	*RPKI*	Quagga*SRx* (virtual) *RTR-BIRD* v1.0	O	*RTR-BIRD* v1.1
Path Validation	*BGPsec*	Quagga*SRx* (virtual)	X	*BDC* *VLR*s

introduced in Section 3 and are eliminated by updating its structure, resulting in *RTR-BIRD* v1.1 that can validate origins without latency.

The Quagga*SRx* was extended for *BGPsec* and was considered the only *BGPsec* router. However, due to the incompleteness and undeployability of *BGPsec*, we could not leverage the implementation for practical *BGPsec* simulations. Instead, we first formulated the computational overhead of *BGPsec* in order to provide quick *BGPsec* analysis as in Section 4. In Section 5, we implemented a comprehensive *BGP*-data analyzing tool called *BDC*, based on which the load reduction algorithms in validating *BGP* paths are proposed in Section 6. As a result, we contributed to secure *BGP* by offering the *Origin Validation* and *Path Validation* based on *PKI* (i.e., strong validation), as well as within a shorter time.

The shortage of manpower and hardware resources might restrict our research. For example, we could not implement a *BGPsec*-capable *BGP* software router. It was because researchers who developed *RTRlib* had quickly completed the development of their *BGPsec*-capable *BIRD* software router although I, as an individual developer, initiated the development much earlier than them. It was the experience of the limitation in developing an overloaded implementation as a single researcher. We somewhat experienced similar limitations in hardware resources. Thereby, our studies highly depend on the enhancement of *BGP* solutions (e.g., a software router, *RPKI*, or *BGPsec*) and the analysis of *BGP* data (e.g., *BGP Update* messages or *RIB* entries), rather than the behavior of actual *BGP* networks (i.e., *BGP* dynamics or real-world routing). As an example, the quantitative analysis such as measuring the

BGP convergence time can be accomplished by injecting announcements and withdrawals of *IP Prefix*es from the global routing domain. However, it is also hard to attain a *BGP* session with a real-world *BGP* speaker for research purposes (i.e., without involving in a monetary relationship). We hope that this paper can be a guide for those who are examining a network protocol in the similar availability in resources.

Bibliography

[1] Y. Rekhter, T. Li, and S. Hares, "RFC 4271 - A Border Gateway Protocol 4 (BGP-4)," Jan. 2006. [Online]. Available: https://tools.ietf.org/html/rfc4271

[2] M. Lepinski, "draft-ietf-sidr-bgpsec-protocol-11 - BGPsec Protocol Specification," Jan. 2015. [Online]. Available: http://tools.ietf.org/html/draft-ietf-sidr-bgpsec-protocol-11

[3] Dave Evans, "The Internet of Things How the Next Evolution of the Internet Is Changing Everything," Cisco, White Paper, Apr. 2011. [Online]. Available: http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[4] Karen Tillman, "How Many Internet Connections are in the World? Right. Now." Jul. 2013. [Online]. Available: http://blogs.cisco.com/news/cisco-connections-counter

[5] Advanced Network Technology Center, University of Oregon, "Route Views Project Page," Jan. 2015. [Online]. Available: http://www.routeviews.org/

[6] J. Hawkinson and T. Bates, "RFC 1930 - Guidelines for creation, selection, and registration of an Autonomous System (AS)," Mar. 1996. [Online]. Available: http://tools.ietf.org/html/rfc1930

[7] V. Fuller and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan," Aug. 2006. [Online]. Available: https://tools.ietf.org/html/rfc4632

[8] IANA, "IANA - Number Resources," Mar. 2016. [Online]. Available: https://www.iana.org/numbers

[9] Z. Morley Mao, D. Johnson, J. Rexford, Jia Wang, and R. Katz, "Scalable and accurate identification of AS-level forwarding paths," vol. 3. IEEE, 2004, pp. 1605–1615. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1354573

[10] M. Winther, "Tier 1 isps: What they are and why they are important," *IDC White Paper*, 2006.

[11] P. Faratin, D. D. Clark, S. Bauer, W. Lehr, P. W. Gilmore, and A. Berger, "The growing complexity of Internet interconnection," *Communications & strategies*, no. 72, p. 51, 2008.

[12] IANA, "Border Gateway Protocol (BGP) Parameters," May 2015. [Online]. Available: http://www.iana.org/assignments/bgp-parameters/bgp-parameters.xhtml

[13] Cisco, "BGP Case Studies (BGP Next Hop Attribute)," Mar. 2016. [Online]. Available: http://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/26634-bgp-toc.html#bgpnexthop

[14] Y. Rekhter and S. R. Sangli, "BGP Extended Communities Attribute," Feb. 2006. [Online]. Available: https://tools.ietf.org/html/rfc4360

[15] Cisco, "BGP Best Path Selection Algorithm," Jul. 2014. [Online]. Available: http://cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13753-25.html

[16] F. Wang and L. Gao, "On inferring and characterizing Internet routing policies," *Journal of Communications and Networks*, vol. 9, no. 4, pp. 350–355, Dec. 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6182869

[17] L. Gao and J. Rexford, "Stable Internet routing without global coordination," *ACM SIGMETRICS Performance Evaluation Review*, vol. 28, no. 1, pp. 307–317, Jun. 2000. [Online]. Available: http://portal.acm.org/citation.cfm?doid=345063.339426

[18] M. Caesar and J. Rexford, "BGP routing policies in ISP networks," *IEEE Network*, vol. 19, no. 6, pp. 5–11, Nov. 2005. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1541715

[19] M. Zhao, S. W. Smith, and D. M. Nicol, "Evaluating the Performance Impact of PKI on BGP Security," in *In Proceedings of 4th Annual PKI Research Workshop (PKI'05*, 2005.

[20] M. Zhao, S. Smith, and D. Nicol, "The performance impact of BPG security," *IEEE Network*, vol. 19, no. 6, pp. 42–48, Nov. 2005. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1541720

[21] R. Chandra, P. Traina, and T. Li, "RFC 1997 - BGP Communities Attribute," Aug. 1996. [Online]. Available: https://tools.ietf.org/html/rfc1997

[22] R. Bush and R. Austein, "RFC 6810 - The Resource Public Key Infrastructure (RPKI) to Router Protocol," Jan. 2013. [Online]. Available: http://tools.ietf.org/html/rfc6810

[23] S. Murphy, "RFC 4272 - BGP Security Vulnerabilities Analysis," Jan. 2006. [Online]. Available: http://tools.ietf.org/html/rfc4272

[24] Dario Vieira, "A Survey of BGP Session Maintenance Issues and Solutions," *Network Protocols & Algorithms*, vol. 2, no. 1, p. 132, Mar. 2010.

[25] A. H. <ahh@cisco.com>, "Protection of BGP Sessions via the TCP MD5 Signature Option," Aug. 1998. [Online]. Available: https://tools.ietf.org/html/rfc2385

[26] Alexander Sotirov, Marc Stevens, Jacob Appelbaum, Arjen Lenstra, David Molnar, Dag Arne Osvik, and Benne de Weger, "MD5 considered harmful today," Jun. 2011. [Online]. Available: http://www.win.tue.nl/hashclash/rogue-ca/

[27] NIST ITL, "NIST.gov - Computer Security Division - Computer Security Resource Center," Mar. 2014. [Online]. Available: http://csrc.nist.gov/groups/ST/hash/policy.html

[28] A. Barbir, S. Murphy, and Y. Yang, "Generic threats to routing protocols," 2006.

[29] B. Christian and T. Tauber, "BGP security requirements," *Work in Progress*, 2008.

[30] R. Mahajan, D. Wetherall, and T. Anderson, "Understanding BGP misconfiguration." ACM Press, 2002, p. 3. [Online]. Available: http://portal.acm.org/citation.cfm?doid=633025.633027

[31] P. Boothe, J. Hiebert, and R. Bush, "How prevalent is prefix hijacking on the Internet," *NANOG36 Talk, February*, 2006.

[32] S. Vissicchio, L. Vanbever, C. Pelsser, L. Cittadini, P. Francois, and O. Bonaventure, "Improving Network Agility With Seamless BGP Reconfigurations," *IEEE/ACM Transactions on Networking*, vol. 21, no. 3, pp. 990–1002, Jun. 2013. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6327628

[33] S. M. Bellovin and E. R. Gansner, "Using link cuts to attack Internet routing," 2003.

[34] M. Lad, X. Zhao, B. Zhang, D. Massey, and L. Zhang, "Analysis of BGP Update Surge during Slammer Worm Attack," in *Distributed Computing - IWDC 2003*, G. Goos, J. Hartmanis, J. van Leeuwen, S. R. Das, and S. K. Das, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, vol. 2918, pp. 66–79. [Online]. Available: http://link.springer.com/10.1007/978-3-540-24604-6_7

[35] Jintae Kim, S. Ko, D. Nicol, X. Dimitropoulos, and G. Riley, "A BGP Attack Against Traffic Engineering," vol. 1. IEEE, 2004, pp. 310–318. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1371332

[36] A. Ramachandran and N. Feamster, "Understanding the network-level behavior of spammers." ACM Press, 2006, p. 291. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1159913.1159947

[37] H. Ballani, P. Francis, and X. Zhang, "A study of prefix hijacking and interception in the internet." ACM Press, 2007, p. 265. [Online]. Available: http://dl.acm.org/citation.cfm?doid=1282380.1282411

[38] A. Pilosov and T. Kapela, "Stealing the Internet: An Internet-scale man in the middle attack," *NANOG-44, Los Angeles, October*, pp. 12–15, 2008. [Online]. Available: https://www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-pilosov-kapela.pdf

[39] K. Zetter, "Revealed: The internet's biggest security hole," *Wired Magazine-ThreadLevel, Aug*, 2008. [Online]. Available: http://www.wired.com/2008/08/revealed-the-in/

[40] D. Montgomery and S. Murphy, "Toward Secure Routing Infrastructures," *IEEE Security & Privacy Magazine*, vol. 4, no. 5, pp. 84–87, Sep. 2006. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1704792

[41] C. Zheng, L. Ji, D. Pei, J. Wang, and P. Francis, "A light-weight distributed scheme for detecting ip prefix hijacks in real-time." ACM Press, 2007, p. 277. [Online]. Available: http://dl.acm.org/citation.cfm?doid=1282380.1282412

[42] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush, "iSPY: Detecting IP Prefix Hijacking on My Own," *IEEE/ACM Transactions on Networking*, vol. 18, no. 6, pp. 1815–1828, Dec. 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5582115

[43] N. Leavitt, "IPv6: Any Closer to Adoption?" *Computer*, vol. 44, no. 9, pp. 14–16, Sep. 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6017167

[44] R. Barrett, S. Haar, and R. Whitestone, "Routing snafu causes Internet outage." Apr. 1997. [Online]. Available: http://citeseerx.ist.psu.edu/showciting;jsessionid=2E4FE73B4D7FB05E73E53062377EC256?cid=651565

[45] Vicent J. Bono, "7007 Explanation and Apology," Apr. 1997. [Online]. Available: http://www.merit.edu/mail.archives/nanog/1997-04/msg00444.html

[46] Tao Wan and P. van Oorschot, "Analysis of BGP prefix origins during Google's May 2005 outage." IEEE, 2006, p. 8 pp. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1639679

[47] Todd Underwood, "Con-Ed Steals the 'Net," Feb. 2015. [Online]. Available: http://research.dyn.com/2006/01/coned-steals-the-net/

[48] RIPE NCC, "YouTube Hijacking: A RIPE NCC RIS case study — RIPE Network Coordination Centre," May 2008. [Online]. Available: https://www.ripe.net/internet-coordination/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study

[49] E. Zmijewski, "Threats to Internet routing and global connectivity," in *20th Annual FIRST Conference*, 2008. [Online]. Available: http://research.dyn.com/wp-content/uploads/2013/05/20thAnnualFIRST.pdf

[50] TeleGeography, "Cable cuts disrupt internet in Middle East and India," Jan. 2008. [Online]. Available: https://www.telegeography.com/products/commsupdate/articles/2008/01/31/cable-cuts-disrupt-internet-in-middle-east-and-india/

[51] Earl Zmijewski, "Reckless Driving on the Internet - Dyn Research | The New Home Of Renesys," Feb. 2009. [Online]. Available: http://research.dyn.com/2009/02/the-flap-heard-around-the-world/

[52] Jim Cowie, "China's 18-Minute Mystery," Nov. 2010. [Online]. Available: http://research.dyn.com/2010/11/chinas-18-minute-mystery/

[53] Tom Paseka, "Why Google Went Offline Today and a Bit about How the Internet Works," Nov. 2012. [Online]. Available: https://blog.cloudflare.com/why-google-went-offline-today-and-a-bit-about/

[54] G. Huston, "Leaking Routes | blabs.apnic.net," May 2012. [Online]. Available: https://labs.apnic.net/?p=139

[55] UA Network Research Lab, "Large Route Leaks — Network Research Lab," 2010. [Online]. Available: http://nrl.cs.arizona.edu/projects/lsrl-events-from-2003-to-2009

[56] X. Hu and Z. M. Mao, "Accurate Real-time Identification of IP Prefix Hijacking." IEEE, May 2007, pp. 3–17. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4223210

[57] R. Perlman, "Network layer protocols with byzantine robustness," Ph.D. dissertation, Massachusetts Institute of Technology, 1989. [Online]. Available: http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-429.pdf

[58] A. Flavel, M. Roughan, N. Bean, and O. Maennel, "Modeling BGP Table Fluctuations," in *Managing Traffic Performance in Converged Networks*, L. Mason, T. Drwiega, and J. Yan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, vol. 4516, pp. 141–153. [Online]. Available: http://link.springer.com/10.1007/978-3-540-72990-7_16

[59] N. Kushman, S. Kandula, D. Katabi, and B. M. Maggs, "R-BGP: Staying connected in a connected world." USENIX, 2007.

[60] A. Akella, B. Maggs, S. Seshan, and A. Shaikh, "On the Performance Benefits of Multihoming Route Control," *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 91–104, Feb. 2008. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4444747

[61] J. H. Park, D. Jen, M. Lad, S. Amante, D. McPherson, and L. Zhang, "Investigating Occurrence of Duplicate Updates in BGP Announcements," in *Passive and Active Measurement*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, A. Krishnamurthy, and B. Plattner, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, vol. 6032, pp. 11–20. [Online]. Available: http://link.springer.com/10.1007/978-3-642-12334-4_2

[62] R. Bolla and R. Bruschi, "An open-source platform for distributed Linux Software Routers," *Computer Communications*, vol. 36, no. 4, pp. 396–410, Feb. 2013. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0140366412003878

[63] T. Griffin and G. Wilfong, "Analysis of the MED oscillation problem in BGP." IEEE Comput. Soc, 2002, pp. 90–99. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1181389

[64] V. Van den Schrieck, P. Francois, and O. Bonaventure, "BGP Add-Paths: The Scaling/Performance Tradeoffs," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 8, pp. 1299–1307, Oct. 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5586442

[65] B. Quoitin, C. Pelsser, O. Bonaventure, and S. Uhlig, "A performance evaluation of BGP-based traffic engineering," *International Journal of*

Network Management, vol. 15, no. 3, pp. 177–191, May 2005. [Online]. Available: http://doi.wiley.com/10.1002/nem.559

[66] T. Griffin, F. Shepherd, and G. Wilfong, "The stable paths problem and interdomain routing," *IEEE/ACM Transactions on Networking*, vol. 10, no. 2, pp. 232–243, Apr. 2002. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=993304

[67] A. Feldmann, O. Maennel, Z. M. Mao, A. Berger, and B. Maggs, "Locating internet routing instabilities," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, p. 205, Oct. 2004. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1030194.1015491

[68] R. Oliveira, B. Zhang, D. Pei, and L. Zhang, "Quantifying Path Exploration in the Internet," *IEEE/ACM Transactions on Networking*, vol. 17, no. 2, pp. 445–458, Apr. 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4814867

[69] A. Sapegin and S. Uhlig, "On the extent of correlation in BGP updates in the Internet and what it tells us about locality of BGP routing events," *Computer Communications*, vol. 36, no. 15-16, pp. 1592–1605, Sep. 2013. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0140366413001898

[70] Q. Li, M. Xu, J. Wu, P. P. Lee, and D. M. Chiu, "Toward a practical approach for BGP stability with root cause check," *Journal of Parallel and Distributed Computing*, vol. 71, no. 8, pp. 1098–1110, Aug. 2011. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0743731511000840

[71] S. Secci, J.-L. Rougier, A. Pattavina, F. Patrone, and G. Maier, "Peering Equilibrium Multipath Routing: A Game Theory Framework for Internet Peering Settlements," *IEEE/ACM Transactions on Networking*, vol. 19, no. 2, pp. 419–432, Apr. 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5556027

[72] W.-H. Hsu and Y.-P. Shieh, "Simple path diversity algorithm for interdomain routing," *IET Communications*, vol. 5, no. 16, pp. 2310–2316, Nov. 2011. [Online]. Available: http://digital-library.theiet.org/content/journals/10.1049/iet-com.2011.0050

[73] J. M. Camacho, A. García-Martínez, M. Bagnulo, and F. Valera, "BGP-XM: BGP eXtended Multipath for transit Autonomous Systems," *Computer Networks*, vol. 57, no. 4, pp. 954–975, Mar. 2013. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1389128612003957

[74] Z. M. Mao, R. Govindan, G. Varghese, and R. H. Katz, "Route flap damping exacerbates internet routing convergence," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 4, p. 221, Oct. 2002. [Online]. Available: http://portal.acm.org/citation.cfm?doid=964725.633047

[75] K. Sriram, D. Montgomery, O. Borchert, O. Kim, and D. Kuhn, "Study of BGP Peering Session Attacks and Their Impacts on Routing Performance," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 10, pp. 1901–1915, Oct. 2006. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1705621

[76] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian, "Delayed Internet routing convergence." ACM Press, 2000, pp. 175–187. [Online]. Available: http://portal.acm.org/citation.cfm?doid=347059.347428

[77] C. T. Ee, V. Ramachandran, B.-G. Chun, K. Lakshminarayanan, and S. Shenker, "Resolving inter-domain policy disputes," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 4, p. 157, Oct. 2007. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1282427.1282399

[78] S. Yilmaz and I. Matta, "An Adaptive Management Approach to Resolving Policy Conflicts," in *NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet*, I. F. Akyildiz, R. Sivakumar, E. Ekici, J. C. d. Oliveira, and J. McNair, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, vol. 4479, pp. 820–831. [Online]. Available: http://link.springer.com/10.1007/978-3-540-72606-7_70

[79] L. Wang, M. Saranu, J. M. Gottlieb, and D. Pei, "Understanding BGP Session Failures in a Large ISP." IEEE, 2007, pp. 348–356. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4215630

[80] D. Obradovic, "Real-time model and convergence time of BGP," vol. 2. IEEE, 2002, pp. 893–901. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1019336

[81] D. Pei, M. Azuma, D. Massey, and L. Zhang, "BGP-RCN: improving BGP convergence through root cause notification," *Computer Networks*, vol. 48, no. 2, pp. 175–194, Jun. 2005. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1389128604003135

[82] C. Labovitz, A. Ahuja, and F. Jahanian, "Experimental study of Internet stability and backbone failures." IEEE Comput. Soc, 1999, pp. 278–285. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=781062

[83] A. Shaikh, A. Varma, L. Kalampoukas, and R. Dube, "Routing stability in congested networks: experimentation and analysis." ACM Press, 2000, pp. 163–174. [Online]. Available: http://portal.acm.org/citation.cfm?doid=347059.347426

[84] N. Feamster, J. Borkenhagen, and J. Rexford, "Guidelines for interdomain traffic engineering," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 5, p. 19, Oct. 2003. [Online]. Available: http://portal.acm.org/citation.cfm?doid=963985.963988

[85] T. Leighton, "Improving performance on the internet," *Communications of the ACM*, vol. 52, no. 2, p. 44, Feb. 2009. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1461928.1461944

[86] K. Sriram, P. Gleichmann, Y.-T. Kim, and D. Montgomery, "Enhanced Efficiency of Mapping Distribution Protocols in Scalable Routing and Addressing Architectures," in *Computer Communications and Networks (ICCCN), 2010 Proceedings of 19th International Conference on*. IEEE, 2010, pp. 1–7.

[87] Z. M. Mao, R. Bush, T. G. Griffin, and M. Roughan, "BGP beacons." ACM Press, 2003, p. 1. [Online]. Available: http://portal.acm.org/citation.cfm?doid=948205.948207

[88] M. Lad, Lixia Zhang, and D. Massey, "Link-Rank: a graphical tool for capturing BGP routing dynamics." IEEE, 2004, pp. 627–640. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1317749

[89] R. Cohen and D. Raz, "The Internet Dark Matter - on the Missing Links in the AS Connectivity Map." IEEE, 2006, pp. 1–12. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4146887

[90] O. Bonaventure, C. Filsfils, and P. Francois, "Achieving Sub-50 Milliseconds Recovery Upon BGP Peering Link Failures," *IEEE/ACM Transactions on Networking*, vol. 15, no. 5, pp. 1123–1135, Oct. 2007. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4346538

[91] B. Augustin, B. Krishnamurthy, and W. Willinger, "IXPs: mapped?" ACM Press, 2009, p. 336. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1644893.1644934

[92] B. Zhang, V. Kambhampati, M. Lad, D. Massey, and L. Zhang, "Identifying BGP routing table transfers." ACM Press, 2005, p. 213. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1080173.1080188

[93] R. Oliveira, Dan Pei, W. Willinger, Beichuan Zhang, and Lixia Zhang, "The (In)Completeness of the Observed Internet AS-level Structure," *IEEE/ACM Transactions on Networking*, vol. 18, no. 1, pp. 109–122, Feb. 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5200324

[94] P. Marchetta, P. Merindol, B. Donnet, A. Pescape, and J.-J. Pansiot, "Topology Discovery at the Router Level: A New Hybrid Tool Targeting ISP Networks," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1776–1787, Oct. 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6027860

[95] R. Oliveira, M. Lad, B. Zhang, and L. Zhang, "Geographically Informed Inter-Domain Routing." IEEE, Oct. 2007, pp. 103–112. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4375841

[96] T. Li, Y. Zhu, K. Xu, and M. Chen, "Performance model and evaluation on geographic-based routing," *Computer Communications*, vol. 32, no. 2, pp. 343–348, Feb. 2009. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0140366408005690

[97] Yihua He, G. Siganos, M. Faloutsos, and S. Krishnamurthy, "Lord of the Links: A Framework for Discovering Missing Links in the Internet Topology," *IEEE/ACM Transactions on Networking*, vol. 17, no. 2, pp. 391–404, Apr. 2009. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4569867

[98] K. Chen, D. R. Choffnes, R. Potharaju, Y. Chen, F. E. Bustamante, D. Pei, and Y. Zhao, "Where the sidewalk ends: extending the internet as graph using traceroutes from P2p users." ACM Press, 2009, p. 217. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1658939.1658964

[99] Department of Homeland Security (DHS), "National Strategy to Secure Cyberspace | Homeland Security," Jul. 2014. [Online]. Available: http://www.dhs.gov/national-strategy-secure-cyberspace

[100] Internet Engineering Task Force (IETF), "Secure Inter-Domain Routing (sidr) - Charter," Dec. 2015. [Online]. Available: http://datatracker.ietf.org/wg/sidr/charter/

[101] NANOG, "Meet us in San Francisco, CA for NANOG 64! | North American Network Operators Group," Jun. 2009. [Online]. Available: https://www.nanog.org/

[102] K. Butler, T. Farley, P. McDaniel, and J. Rexford, "A Survey of BGP Security Issues and Solutions," *Proceedings of the IEEE*,

vol. 98, no. 1, pp. 100–122, Jan. 2010. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5357585

[103] X. Zhao, D. Pei, L. Wang, D. Massey, A. Mankin, S. F. Wu, and L. Zhang, "An analysis of BGP multiple origin AS (MOAS) conflicts." ACM Press, 2001, p. 31. [Online]. Available: http://portal.acm.org/citation.cfm?doid=505202.505207

[104] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. F. Wu, and L. Zhang, "Protecting BGP Routes to Top Level DNS Servers," in *Proceedings of the 23rd International Conference on Distributed Computing Systems*, ser. ICDCS '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 322–. [Online]. Available: http://dl.acm.org/citation.cfm?id=850929.851984

[105] Xiaoliang Zhao, Dan Pei, Lan Wang, D. Massey, A. Mankin, S. Wu, and Lixia Zhang, "Detection of invalid routing announcement in the Internet." IEEE Comput. Soc, 2002, pp. 59–68. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1028887

[106] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang, "PHAS: A Prefix Hijack Alert System," in *Proceedings of the 15th Conference on USENIX Security Symposium - Volume 15*, ser. USENIX-SS'06. Berkeley, CA, USA: USENIX Association, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267336.1267347

[107] L. Subramanian, V. Roth, I. Stoica, S. Shenker, and R. Katz, "Listen and whisper: Security mechanisms for BGP," in *Proc. First Symposium on Networked Systems Design and Implementation (NSDI)*, 2004, p. 11.

[108] J. Karlin, S. Forrest, and J. Rexford, "Pretty Good BGP: Improving BGP by Cautiously Adopting Routes." IEEE, Nov. 2006, pp. 290–299. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4110301

[109] M. Zhang, B. Liu, and B. Zhang, "Safeguarding Data Delivery by Decoupling Path Propagation and Adoption." IEEE, Mar. 2010, pp. 1–5. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5462200

[110] O. Nordström and C. Dovrolis, "Beware of BGP attacks," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, p. 1, Apr. 2004. [Online]. Available: http://portal.acm.org/citation.cfm?doid=997150.997152

[111] Ke Zhang, Xiaoliang Zhao, and S. Wu, "An analysis on selective dropping attack in BGP." IEEE, 2004, pp. 593–599. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1395106

[112] J. Chandrashokar, Zhenhai Duan, Zhi-Li Zhang, and J. Krasky, "Limiting path exploration in GBP," vol. 4. IEEE, 2005, pp. 2337–2348. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1498520

[113] S. Kent, C. Lynn, J. Mikkelson, and K. Seo, "Secure Border Gateway Protocol (S-BGP)- real world performance and deployment issues," in *Proceedings of Network and Distributed System and Security Symposium2000 (NDSS'00)*, 2000.

[114] S. Goldberg, M. Schapira, P. Hummon, and J. Rexford, "How secure are secure interdomain routing protocols," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, p. 87, Aug. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?doid=1851275.1851195

[115] A. Boldyreva and R. Lychev, "Provable security of S-BGP and other path vector protocols: model, analysis and extensions." ACM Press, 2012, p. 541. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2382196.2382254

[116] Y. Xiang, Z. Wang, J. Wu, X. Shi, and X. Yin, "Sign What You Really Care about – Secure BGP AS Paths Efficiently," in *NETWORKING 2012*, D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, R. Bestak, L. Kencl, L. E. Li, J. Widmer, and H. Yin, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, vol. 7289, pp. 259–273. [Online]. Available: http://link.springer.com/10.1007/978-3-642-30045-5_20

[117] P. v. Oorschot, T. Wan, and E. Kranakis, "On interdomain routing security and pretty secure BGP (psBGP)," *ACM Transactions on Information and System Security*, vol. 10, no. 3, pp. 11–es, Jul. 2007. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1266977.1266980

[118] S. M. Bellovin, J. Ioannidis, and R. Bush, "Position paper: Operational requirements for secured BGP," in *DHS Secure Routing Workshop*. Citeseer, 2005.

[119] P. Zhu, H. Cao, L. T. Yang, and K. Chen, "AS Alliance based security enhancement for inter-domain routing protocol," *Mathematical and Computer Modelling*, vol. 55, no. 1-2, pp. 241–255, Jan.

2012. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S089571771100344X

[120] R. Lychev, S. Goldberg, and M. Schapira, "BGP security in partial deployment: is the juice worth the squeeze?" ACM Press, 2013, p. 171. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2486001.2486010

[121] C. Labovitz, A. Ahuja, R. Wattenhofer, and S. Venkatachary, "The Impact of Internet Policy and Topology on Delayed Routing Convergence," 2001, pp. 537–546.

[122] Geoff Huston, "AS6447 - BGP Table Statistics," Feb. 2015. [Online]. Available: http://bgp.potaroo.net/as6447/

[123] S. Kent, C. Lynn, and K. Seo, "Secure Border Gateway Protocol (S-BGP)," *IEEE Journal on Selected Areas in Communications*, vol. 18, no. 4, pp. 582–592, Apr. 2000. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=839934

[124] M. Yannuzzi, X. Masip-Bruin, and O. Bonaventure, "Open issues in interdomain routing: a survey," *IEEE Network*, vol. 19, no. 6, pp. 49–56, Nov. 2005. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1541721

[125] CIDR, "AS2.0 BGP Table Statistics," Mar. 2016. [Online]. Available: http://www.cidr-report.org/cgi-bin/plota?file=%2fvar%2fdata%2fbgp%2fas2.0%2fbgp%2dactive%2etxt&descr=Active%20BGP%20entries%20%28FIB%29&ylabel=Active%20BGP%20entries%20%28FIB%29&with=step

[126] Russ White, "Securing BGP Through Secure Origin BGP - The Internet Protocol Journal - Volume 6, Number 3 - Cisco Systems," 2003. [Online]. Available: http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_6-3/securing_bgp_sobgp.html

[127] R. Bush and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol," Jan. 2013. [Online]. Available: https://tools.ietf.org/html/rfc6810

[128] M. Lepinski, S. Kent, and D. Kong, "RFC 6482 - A Profile for Route Origin Authorizations (ROAs)," Feb. 2012. [Online]. Available: http://tools.ietf.org/html/rfc6482

[129] G. Huston and G. Michaelson, "RFC 6483 - Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs)," Feb. 2012. [Online]. Available: http://tools.ietf.org/html/rfc6483

[130] C. Lynn, S. Kent, and K. Seo, "RFC 3779 - X.509 Extensions for IP Addresses and AS Identifiers," Jun. 2004. [Online]. Available: http://tools.ietf.org/html/rfc3779

[131] NIST ITL Advanced Network Technologies Division, "RPKI Deployment Monitor," Mar. 2015. [Online]. Available: http://rpki-monitor.antd.nist.gov/

[132] P. Jakma, V. Jardin, D. Lamparter, and A. Schorr, "Quagga Software Routing Suite," Jun. 2015. [Online]. Available: http://www.nongnu.org/quagga/

[133] CZ.NIC Labs, "The BIRD Internet Routing Daemon Project," May 2013. [Online]. Available: http://bird.network.cz/

[134] XORP Team, "Welcome to XORP," Dec. 2015. [Online]. Available: http://www.xorp.org/

[135] Vyatta, "Virtual Router - Brocade Vyatta 5400 vRouter Overview," Dec. 2015. [Online]. Available: http://www.brocade.com/products/all/network-functions-virtualization/product-details/5400-vrouter/index.page

[136] NIST ITL, "BGP Secure Routing Extension (BGP-SRx)," Apr. 2014. [Online]. Available: http://www-x.antd.nist.gov/bgpsrx/

[137] O. Filip, L. Forst, P. Machek, M. Mares, and O. Zajicek, "BIRD internet routing daemon," *NANOG-48, Austin, TX*, 2010.

[138] INET Research Group, "RTRlib," 2015. [Online]. Available: http://rtrlib.realmv6.org/

[139] M. Wählisch, F. Holler, T. C. Schmidt, and J. H. Schiller, "Updates from the internet backbone: An rpki/rtr router implementation, measurements, and analysis," in *NDSS*, 2013.

[140] G. Huston, M. Rossi, and G. Armitage, "Securing BGP - A Literature Survey," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 2, pp. 199–222, 2011. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5473881

[141] D. M. Nicol, S. W. Smith, and M. Zhao, "Evaluation of efficient security for BGP route announcements using parallel simulation," *Simulation Modelling Practice and Theory*, vol. 12, no. 3-4, pp. 187–216, Jul. 2004. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S1569190X04000383

[142] M. Lepinski, R. Barnes, and S. Kent, "An Infrastructure to Support Secure Internet Routing," Feb. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6480

[143] S. Kent, C. Lynn, and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers," Jun. 2004. [Online]. Available: http://tools.ietf.org/html/rfc3779

[144] M. Lepinski, D. Kong, and S. Kent, "A Profile for Route Origin Authorizations (ROAs)," Feb. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6482

[145] M. Lepinski, A. Chi, and S. Kent, "Signed Object Template for the Resource Public Key Infrastructure (RPKI)," Feb. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6488

[146] S. Bellovin, R. Bush, and D. Ward, "draft-ietf-sidr-bgpsec-reqs-12 - Security Requirements for BGP Path Validation," Jul. 2014. [Online]. Available: http://tools.ietf.org/html/draft-ietf-sidr-bgpsec-reqs-12

[147] M. Lepinski and S. Turner, "draft-ietf-sidr-bgpsec-overview-06 - An Overview of BGPsec," Jan. 2015. [Online]. Available: http://tools.ietf.org/html/draft-ietf-sidr-bgpsec-overview-06

[148] J. Scudder and R. Chandra, "Capabilities Advertisement with BGP-4," Feb. 2009. [Online]. Available: http://tools.ietf.org/html/rfc5492

[149] P. FIPS, "180-4," *Federal Information Processing Standards Publication, Secure Hash*, 2012. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf

[150] G. H. <gih@apnic.net>, "The Profile for Algorithms and Key Sizes for Use in the Resource Public Key Infrastructure (RPKI)," Feb. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6485

[151] S. T. <turners@ieca.com>, "BGP Algorithms, Key Formats, & Signature Formats," Jan. 2015. [Online]. Available: https://tools.ietf.org/html/draft-ietf-sidr-bgpsec-algs-09

[152] K. Kim, I.-h. Jang, and Y. Kim, "Evaluating the performance impact of RTR-BIRD in origin validation." 7800 York Rd., Towson, MD, USA: ACM Press, 2014, pp. 198–203. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2663761.2664202

[153] Hurricane Electric Internet Services, "BGP Peer Report - bgp.he.net," Mar. 2015. [Online]. Available: http://bgp.he.net/report/peers

[154] Patrick Maigron, "Regional Internet Registries Statistics," Feb. 2015. [Online]. Available: http://www-public.it-sudparis.eu/~maigron/RIR_Stats/index.html

[155] L. Blunk, M. Karir, and C. Labovitz, "RFC 6396 - Multi-Threaded Routing Toolkit (MRT) Routing Information Export Format," Oct. 2011. [Online]. Available: https://tools.ietf.org/html/rfc6396

[156] GNS3, "GNS3," Jul. 2015. [Online]. Available: http://www.gns3.com/

[157] VINT Project Team, "The Network Simulator - ns-2," Jul. 2015. [Online]. Available: http://www.isi.edu/nsnam/ns/

[158] Savannah Project Team, "Quagga Software Routing Suite," Mar. 2013. [Online]. Available: http://www.nongnu.org/quagga/

[159] TIS Labs, "BIRD BGPsec Implementation," Mar. 2015. [Online]. Available: http://bgpsec.tislabs.com/

[160] Computer Networks Research Group at Roma Tre University, "BGPlay – graphical visualisation of BGP updates," Jul. 2015. [Online]. Available: http://bgplay.routeviews.org/

[161] Colorado State University, "BGPmon - Home," Jan. 2015. [Online]. Available: http://www.bgpmon.io/index.html

[162] C. Hu and K. Chen, "Border gateway protocol monitoring system can be cost effective," *IET Communications*, vol. 5, no. 15, pp. 2231–2240, Oct. 2011. [Online]. Available: http://digital-library.theiet.org/content/journals/10.1049/iet-com.2010.0432

[163] Tony Bates, Philip Smith, and Geoff Huston, "CIDR Report," Feb. 2015. [Online]. Available: http://www.cidr-report.org/as2.0/#General_Status

[164] CAIDA, "AS Rank: AS Ranking - CAIDA : http://as-rank.caida.org," Feb. 2015. [Online]. Available: http://as-rank.caida.org/?mode0=as-ranking

[165] "IP Tools - ASN lookup and information," Jul. 2014. [Online]. Available: https://www.ultratools.com/tools/asnInfo

[166] RIPE NCC, "RIPEstat — RIPEstat Data API," Oct. 2015. [Online]. Available: https://stat.ripe.net/docs/data_api#BGPUpdates

[167] J. Wu, Z. M. Mao, J. Rexford, and J. Wang, "Finding a needle in a haystack: Pinpointing significant BGP routing changes in an IP network," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 2005, pp. 1–14.

[168] K. Sriram, O. Borchert, O. Kim, P. Gleichmann, and D. Montgomery, "A Comparative Analysis of BGP Anomaly Detection and Robustness Algorithms." IEEE, Mar. 2009, pp. 25–38. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4804424

[169] E. Gregori, A. Improta, L. Lenzini, L. Rossi, and L. Sani, "Inferring geography from BGP raw data." IEEE, Mar. 2012, pp. 208–213. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6193493

[170] T. Mangin, "ExaBGP - A new Tool to Interact with BGP — RIPE Labs," Jul. 2010. [Online]. Available: https://labs.ripe.net/Members/thomas_mangin/content-exabgp-new-tool-interact-bgp

[171] The Trac open source project, "RPKI project tracker," Feb. 2014. [Online]. Available: http://rpki.net/wiki

[172] L. Blunk, M. Karir, and C. Labovitz, "RFC 6396 - Multi-Threaded Routing Toolkit (MRT) Routing Information Export Format," Oct. 2011. [Online]. Available: http://tools.ietf.org/html/rfc6396

[173] B. Fiorelli, "RIPE Database REST API Documentation — RIPE Labs," Feb. 2011. [Online]. Available: https://labs.ripe.net/ripe-database/database-api/api-documentation

[174] Information Sciences Institute University of Southern California, "INTERNET PROTOCOL," Sep. 1981. [Online]. Available: https://www.ietf.org/rfc/rfc791.txt

[175] RIPE NCC, "RPKI Validator API — RIPE Network Coordination Centre," May 2015. [Online]. Available: https://www.ripe.net/support/documentation/developer-documentation/rpki-validator-api/rpki-validator-api

[176] G. Huston, S. Weiler, G. Michaelson, and S. Kent, "Resource Public Key Infrastructure (RPKI) Trust Anchor Locator," Feb. 2012. [Online]. Available: https://tools.ietf.org/html/rfc6490

[177] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "BGP routing stability of popular destinations." ACM Press, 2002, p. 197. [Online]. Available: http://portal.acm.org/citation.cfm?doid=637201.637232

[178] Lixin Gao, "On inferring autonomous system relationships in the Internet," *IEEE/ACM Transactions on Networking*, vol. 9, no. 6, pp. 733–745, Dec. 2001. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=974527

[179] IANA, "Autonomous System (AS) Numbers," Nov. 2015. [Online]. Available: http://www.iana.org/assignments/as-numbers/as-numbers.xhtml

[180] E. Gregori, A. Improta, L. Lenzini, L. Rossi, and L. Sani, "On the incompleteness of the AS-level graph: a novel methodology for BGP route collector placement." ACM Press, 2012, p. 253. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2398776.2398803

[181] Carolyn Duffy Marsan, "Six worst Internet routing attacks," Jan. 2009. [Online]. Available: http://www.networkworld.com/article/2272520/lan-wan/six-worst-internet-routing-attacks.html

Appendix A

Curriculum Vita

NAME: Kyoungha Kim

[redacted]

PROGRAM OF STUDY: Information Technology

DEGREE AND DATE TO BE CONFERRED: Doctoral of Science (D.Sc.), May 2016

PARTICULARS

EDUCATION

Towson University — Towson, MD

Ph. D. in Computer Science — *2012-, Graduating in May 2016*

Towson University — Towson, MD

M. S. in Computer Science — *January 2012*

Distinction in Research

Shinhan University, Gageum-ro, Uijeongbu-si — Gyeonggi-do, Korea

B. Sc. in Computer Science — *February 2009*

PROFESSIONAL PUBLICATIONS

PROCEEDINGS

1. Kim, Kyoungha, and YanggonKim. "The security appliance to BIRD software router." Proceedings of the 8th International Conference on Ubiquitous Information Management and Communication. ACM, 2014. (short-listed)

2. Kim, Kyoungha, Ik-hyeonJang, and YanggonKim. "Evaluating the performance impact of RTR-BIRD in origin validation." Proceedings of the 2014 Conference on Research in Adaptive and Convergent Systems. ACM, 2014.

3. Kim, Kyoungha, and YanggonKim. "Comparative analysis on the signature algorithms to validate AS paths in BGPsec." Computer and Information Science (ICIS), 2015 IEEE/ACIS 14th International Conference on. IEEE, 2015.