Terakreditasi SINTA Peringkat 2
Surat Keputusan Dirjen Pendidikan Tinggi, Riset dan Teknologi No. 158/E/KPT/2021
masa berlaku mulai Vol. 5 No. 2 tahun 2021 s.d Vol. 10 No.1 tahun 2026

Terbit online pada laman web jurnal: **http://jurnal.iaii.or.id**

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. 5 No. 6 (2021) 1161 – 1170 ISSN Media Elektronik: 2580-0760

Implementasi BGP dan *Resource Public Key Infrastructure* menggunakan BIRD untuk Keamanan *Routing*

Valen Brata Pranaya[1], Theophilus Wellem[2]
[1,2]Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana
[1]672018072@student.uksw.edu, [2]theophilus.wellem@uksw.edu*

Abstract

The validity of the routing advertisements sent by one router to another is essential for Internet connectivity. To perform routing exchanges between Autonomous Systems (AS) on the Internet, a protocol known as the Border Gateway Protocol (BGP) is used. One of the most common attacks on routers running BGP is prefix hijacking. This attack aims to disrupt connections between AS and divert routing to destinations that are not appropriate for crimes, such as fraud and data breach. One of the methods developed to prevent prefix hijacking is the Resource Public Key Infrastructure (RPKI). RPKI is a public key infrastructure (PKI) developed for BGP routing security on the Internet and can be used by routers to validate routing advertisements sent by their BGP peers. RPKI utilizes a digital certificate issued by the Certification Authority (CA) to validate the subnet in a routing advertisement. This study aims to implement BGP and RPKI using the Bird Internet Routing Daemon (BIRD). Simulation and implementation are carried out using the GNS3 simulator and a server that acts as the RPKI validator. Experiments were conducted using 4 AS, 7 routers, 1 server for BIRD, and 1 server for validators, and there were 26 invalid or unknown subnets advertised by 2 routers in the simulated topology. The experiment results show that the router can successfully validated the routing advertisement received from its BGP peer using RPKI. All invalid and unknown subnets are not forwarded to other routers in the AS where they are located such that route hijacking is prevented.

Keywords: Border Gateway Protocol, Resource Public Key Infrastructure, Bird Internet Routing Daemon, RPKI Validator.

Abstrak

Validitas dari *routing advertisement* yang dikirimkan oleh suatu *router* ke *router* lainnya merupakan hal yang penting untuk konektivitas Internet. Untuk melakukan pertukaran *routing* antar *Autonomous System* (AS) di Internet, digunakan suatu protokol yang dikenal sebagai *Border Gateway Protocol* (BGP). Salah satu serangan yang umum terjadi pada *router* yang menjalankan BGP adalah *prefix hijacking*. Serangan ini bertujuan untuk mengacaukan koneksi antar AS dan mengalihkan *routing* menuju tujuan yang tidak semestinya untuk tindak kejahatan, misalnya *fraud* dan pencurian data. Salah satu cara yang dikembangkan untuk mencegah *prefix hijacking* adalah *Resource Public Key Infrastructure* (RPKI). RPKI merupakan suatu *public key infrastructure* (PKI) yang dikembangkan untuk keamanan BGP *routing* pada Internet dan dapat digunakan oleh *router* untuk melakukan validasi terhadap *routing advertisement* yang dikirimkan oleh BGP *peer*-nya. RPKI memanfaatkan sertifikat digital yang dikeluarkan oleh *Certification Authority* (CA) dalam melakukan validasi terhadap *subnet* pada suatu *routing advertisement*. Penelitian ini bertujuan untuk mengimplementasikan BGP dan RPKI menggunakan *Bird Internet Routing Daemon* (BIRD). Simulasi dan implementasi dilakukan menggunakan *simulator* GNS3 dan sebuah *server* yang berperan sebagai RPKI *validator*. Pengujian dilakukan menggunakan 4 AS, 7 router, 1 server untuk BIRD, dan 1 server untuk *validator*, serta terdapat sebanyak 26 *invalid* atau *unknown subnet* yang di-*advertise* oleh 2 *router* pada topologi yang disimulasikan. Dari pengujian yang dilakukan didapatkan hasil bahwa *router* pada AS berhasil melakukan validasi terhadap *routing advertisement* yang diterima dari BGP *peer*-nya. Semua *subnet* yang *invalid* dan *unknown* tidak diteruskan ke *router* lain dalam AS di mana *router* tersebut berada, sehingga kekacauan *routing* dapat dicegah.

Kata kunci: *Border Gateway Protocol, Resource Public Key Infrastructure, Bird Internet Routing Daemon, RPKI Validator.*

Diterima Redaksi: 16-11-2021 | Selesai Revisi: 22-12-2021 | Diterbitkan Online: 30-04-2021

1. Pendahuluan

Border Gateway Protocol (BGP) merupakan protokol yang digunakan untuk pertukaran rute antar *Autonomous System* (AS) atau domain di Internet [1]. Protokol ini mendukung *Classless Inter Domain Routing* (CIDR) dan umumnya diterapkan pada *Internet Service Provider* (ISP) atau organisasi-organisasi lainnya. Suatu AS merupakan sekumpulan *router* yang berada di bawah administrasi dari suatu entitas tertentu (misalnya, ISP) dan menjalankan *routing policy* tertentu. Sebagai identitasnya, setiap AS diberikan nomor yang disebut *Autonomous System Number* (ASN) yang merupakan suatu angka 32-bit. Pemberian ASN ini diatur oleh *Regional Internet Registry* (RIR) misalnya APNIC dan RIPE-NCC. Router-router dalam AS menggunakan *Interior Gateway Protocol* (IGP) untuk melakukan *routing* terhadap paket (IP *packet*) di dalam AS tersebut dan *Border Gateway Protocol* (BGP) untuk melakukan routing ke AS yang lain. Secara umum, *router* dalam suatu AS yang menjalankan BGP adalah *router* yang menjadi penghubung antara AS tersebut dengan AS yang lain.

Router dalam suatu AS dapat melakukan pertukaran rute dari dan ke luar AS tersebut serta mempublikasi atau mengumumkan rute (BGP *announcement* atau *route advertisement*) ke berbagai tujuan (jaringan) yang ada di Internet. Jaringan-jaringan di Internet menggunakan IP *prefix* tertentu sebagai alamatnya sesuai aturan pengalamatan pada *Internet Protocol* (IP). Suatu *prefix* jaringan biasanya dituliskan dalam notasi CIDR misalnya, 195.55.100.0/24 yang berarti 24-bit dari alamat tersebut dialokasikan untuk *network prefix* dan 8-bit sisanya digunakan untuk alamat host yang berada pada jaringan tersebut. *Host* dengan alamat 195.55.100.1 hingga 195.55.100.254 berada pada jaringan ini.

Dengan pengalamatan ini, paket yang dikirim dari suatu komputer (*source host*) dapat sampai ke komputer tujuannya (*destination host*) yang juga berada dalam suatu jaringan (*destination network*). Dalam perjalanannya, paket tersebut dapat melalui berbagai AS sesuai dengan rute yang ditentukan oleh *router-router* yang dilalui paket tersebut dari tempat asal sampai ke tujuannya. BGP merupakan *path vector routing protocol* di mana dalam menentukan rute selalu mengacu kepada *path* terbaik yang didapat dari *router* lainnya melalui proses *routing advertisement* [2]. *Routing table* yang dimiliki oleh suatu *router* yang menjalankan protokol BGP berisikan *prefix* yang diterima dari router BGP lainnya yang menjadi *peer*-nya melalui proses pertukaran *routing table*.

Terdapat beberapa cara untuk menerapkan protokol BGP, salah satunya adalah dengan menggunakan aplikasi *routing* seperti *Quagga* [3] dan *Bird Internet Routing Daemon* (BIRD) [4]. Aplikasi ini dapat dipasang pada komputer yang menggunakan sistem operasi yang termasuk dalam keluarga UNIX (misalnya Linux, FreeBSD, NetBSD) sehingga komputer tersebut dapat digunakan sebagai *router*. *Quagga* dan BIRD mendukung *static routing*, BGP, dan beberapa IGP untuk *dynamic routing* (misalnya, *Open Shortest Path First* (OSPF), *Routing Information Protocol* (RIP)). Salah satu keunggulan dari BIRD adalah konsumsi sumber daya (*resource*) prosesor dan memori yang lebih kecil dibandingkan *Quagga*. Berdasarkan penelitian yang dilakukan oleh Ondrej [5], didapatkan bahwa BIRD mampu memproses *routing* dengan lebih cepat dan menggunakan memori yang lebih kecil. Selain itu, BIRD dapat menerapkan konfigurasi *filter* yang cukup lengkap, salah satunya adalah protokol untuk penerapan *Resource Public Key Infrastructure* (RPKI) [6][7].

Salah satu kelemahan dari protokol BGP adalah bahwa protokol ini dapat diserang menggunakan serangan yang dikenal sebagai *BGP hijacking* atau *prefix hijacking* [8][9]. *Prefix hijacking* adalah keadaan di mana suatu AS mempublikasikan (*announce* atau *advertise*) *prefix* yang tidak diketahui bagaimana cara mencapainya atau mempublikasikan *prefix* yang tidak dimiliki oleh AS tersebut (*false prefix announcement*) [10]. Jika *prefix* palsu yang diumumkan ini lebih spesifik dari *prefix* yang seharusnya, maka *routing* dapat dialihkan ke suatu jaringan (*subnet*) tertentu oleh pihak yang tidak bertanggung jawab. Tujuannya adalah untuk memperoleh informasi secara ilegal atau untuk mengacaukan proses *routing* pada Internet. Perlu diingat bahwa dalam proses *routing*, *router* akan memilih *subnet* dengan *prefix* yang lebih spesifik jika terdapat dua *subnet* yang sama tetapi memiliki *prefix* yang berbeda.

Salah satu cara untuk mencegah BGP *hijacking* adalah dengan menerapkan RPKI yang digunakan untuk membatasi AS mana saja yang dapat melakukan *announcement* terhadap IP *prefix* tertentu [11]. Selain itu, dengan RPKI dapat dilakukan validasi (secara kriptografi) terhadap suatu AS dan *prefix* yang boleh diumumkan oleh AS tersebut. Hal ini digunakan untuk melakukan verifikasi otorisasi terhadap AS yang mengumumkan *prefix* tersebut. Dengan kata lain, router yang menerima suatu BGP *announcement* dapat memeriksa apakah suatu AS memiliki otoritas (*authorized*) untuk mengumumkan rute menuju suatu *prefix* di Internet.

Pada RPKI digunakan X.509 PKI *certificate* yang membuktikan kepemilikan dari IP *prefix* dan nomor AS. Sertifikat pada RPKI dikenal dengan nama *resource certificate* dan sertifikat ini mengikat (*binding*) suatu kunci publik (*public key*) dengan nomor AS dan *IP prefix* [12]. Selain itu, terdapat juga objek (*digitally signed object*) yang dikenal dengan nama *Route Origin Authorization* (ROA) yang memberikan otorisasi kepada suatu AS *prefix* mana saja yang boleh diumumkan oleh AS tersebut [10][13]. Suatu ROA berisi nomor AS yang

DOI: https://doi.org/10.29207/resti.v5i6.3631

diberikan otorisasi, IP *prefix* atau *address space* yang dapat di-*advertise* oleh AS tersebut, dan panjang maksimum dari *prefix* atau *address space* yang dapat di-*advertise*. Secara umum, pemilik sah dari suatu IP *address space* (dalam hal ini *network operator*, ISP, atau organisasi lainnya yang medapatkan IP *address space* dari RIR) membuat ROA yang memberikan otorisasi pada AS untuk mengumumkan atau melakukan *advertisement* dari *prefix* miliknya. ROA ditandatangani menggunakan sertifikat yang dikeluarkan oleh *Certification Authority* (CA) yang dalam hal ini adalah RIR. ROA ini kemudian disimpan dalam *repository* milik RIR yang dapat diakses oleh RPKI *validator*.

Dalam melakukan validasi terhadap *prefix* yang diumumkan oleh suatu *router*, *router* (yang menerima BGP *announcement*) menggunakan proses *filtering* yang dikenal sebagai *Route Origin Validation* (ROV) atau BGP *origin validation* untuk menyaring *prefix* pada *routing table* BGP [13]. Proses validasi ini dilakukan oleh RPKI *validator* yang akan mengirimkan daftar prefix yang telah divalidasi ke router. Komunikasi antara RPKI validator dan router dilakukan menggunakan protokol *RPKI-to-Router* (RTR) [14]. Terdapat tiga hasil dari proses ROV: 1) *valid*, apabila AS dan *prefix* sesuai dengan ROA, 2) *invalid*, apabila AS melakukan *announcement prefix* yang tidak sesuai dengan ROA, dan 3) *not found*, apabila *prefix* tidak ditemukan dalam *database* ROA (atau *Route Validation database*).

Penelitian ini bertujuan untuk mengimplementasikan protokol BGP dan RPKI dalam suatu AS untuk mengamankan AS tersebut dari serangan BGP *hijacking*. Implementasi dilakukan menggunakan BIRD sehingga memudahkan konfigurasi protokol maupun *filtering* menggunakan RPKI. Penelitian dilakukan dengan simulasi menggunakan GNS3 [15]. Dalam simulasi digunakan 2 *router upstream*, 4 *router* lokal, 1 *router* dengan AS *valid*, 1 *routing server* BIRD berbasis *Ubuntu*, dan 1 *server RPKI Validator* yang terhubung secara *remote* melalui Internet.

Penelitian dimulai dengan melakukan kajian terhadap beberapa penelitian terdahulu yang berkaitan. Musril [16] melakukan penelitian mengenai simulasi interkoneksi antar AS menggunakan BGP. Hasil penelitian menunjukkan jika BGP mampu melakukan pertukaran rute dari dan ke luar dari AS dan bekerja berdasarkan *reliable service*, yang artinya setiap sesi komunikasi yang menggunakan protokol tersebut harus dipastikan sampai atau tidaknya data yang dikirimkan pada sesi komunikasi tersebut.

Fathurohman [17] melakukan penelitian untuk implementasi protokol BGP pada jaringan publik Universitas Muhammadiyah Semarang (Unimus). Jaringan publik tersebut terdiri menghubungkan beberapa kampus yang terkoneksi dengan *ICT center* dan memiliki dua koneksi *upstream*. Implementasi dilakukan agar semua fakultas dapat saling terhubung secara *dynamic* dan koneksi terhadap dua ISP pada pusat data Unimus menjadi lancar. Implementasi BGP dilakukan tanpa menyertakan RPKI.

Marcus dan Tfuakani [18] melakukan penelitian untuk merancang jaringan skala besar memakai BGP berbasis *MikroTik*. Tujuan penelitian tersebut adalah untuk meningkatkan layanan Internet melalui manajemen jaringan internal dalam Universitas Merdeka Malang. Hasil studi memberikan gambaran umum untuk membuat jaringan protokol *routing* BGP dan meminimalkan biaya yang sangat mahal karena BGP dapat mengurutkan tujuan pengguna di dalam negeri dan luar negeri dengan jalur berbeda di jaringan yang sama.

Chung, dkk. [10] melakukan studi mengenai implementasi RPKI beserta *Invalid Route Origins* untuk mengetahui sejauh mana implementasi dilakukan. Hasil penelitian menunjukkan bahwa adopsi RPKI terbilang cukup cepat meskipun belum maksimal dan membuktikan jika teknologi RPKI sudah siap digunakan meskipun masih ada beberapa kekurangan yang umumnya disebabkan adanya miskonfigurasi. Secara keseluruhan, RPKI sudah mampu mendeteksi *prefix* yang berbahaya pada studi tersebut.

Sermpezis, dkk. [8] melakukan survei terhadap beberapa operator jaringan dan hasilnya adalah bahwa *prefix hijacking* cukup sering terjadi, dan salah satu penyebabnya adalah karena kebanyakan operator tidak mengimplementasikan RPKI sebagai pertahanan yang bersifat aktif. Sementara hanya sedikit yang mengimplementasikan secara total baik menggunakan ROA dan ROV.

Gilad, dkk. [19] melakukan penelitian dan survei untuk mengungkap tingkat implementasi RPKI yang masih rendah. Hanya sedikit *prefix* yang di-*advertise* oleh BGP yang terdaftar oleh ROA. Selain itu, kebanyakan ISP tidak menerapkan ROV untuk validasi pada RPKI atau dapat dikatakan *partial deployment* sehingga penerapannya menjadi kurang efektif.

Chang, dkk. [20] melakukan penelitian untuk uji coba RPKI pada protokol BGP dalam simulasi. RPKI dibangun berdasarkan sistem PKI yang sudah mapan dan menggunakan teknik kriptografi *public key* untuk memverifikasi identitas melalui *digital certificates*. Dalam sistem RPKI, rute dapat dikategorikan sebagai *valid*, *invalid*, atau *unknown*. RPKI bekerja menggunakan *validator* yang terhubung pada repositori *Regional Internet Registry* (RIR) untuk menentukan proses *routing* BGP.

Testart, dkk. [21] melakukan penelitian untuk mengukur seberapa besar manfaat penerapan dan registrasi RPKI. Hasil penelitian menunjukkan bahwa pendaftaran AS dan *prefix* ke dalam RIR mampu mencegah terjadinya *hijacking*. Dengan semakin banyaknya AS dan *prefix* yang didaftarkan maka peluang untuk melakukan *hijack* menjadi semakin kecil.

DOI: https://doi.org/10.29207/resti.v5i6.3631

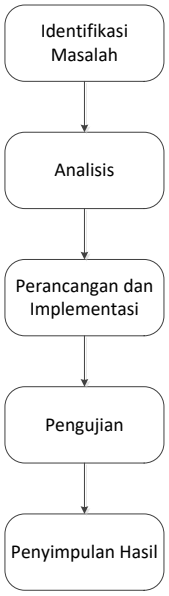
Berbeda dengan penelitian terdahulu, penelitian ini memanfaatkan BIRD yang berjalan pada sistem operasi *Ubuntu* untuk menerapkan protokol BGP dan *filtering* dengan RPKI. Tabel 1 menunjukkan rangkuman beberapa penelitian terdahulu.

Tabel 1. Penelitian Terdahulu

No	Tahun	Penulis	Studi yang dilakukan
1	2016	Chang, dkk.	BGP dan RPKI.
2	2017	Gilad, dkk.	Survei terhadap sejauh mana implementasi RPKI.
3	2017	Musril	Implementasi BGP tanpa RPKI.
4	2018	Sermpezis, dkk.	Survei terhadap operator jaringan mengenai BGP *hijack*.
5	2019	Chung, dkk.	Studi implementasi RPKI.
6	2019	Marcus dan Tfuakani	BGP tanpa RPKI.
7	2020	Testart, dkk.	Mengukur manfaat implementasi RPKI.
8	2021	Fathurohman	Implementasi BGP tanpa RPKI.

2. Metode Penelitian

Tahapan dalam penelitian ini terlihat pada Gambar 1.

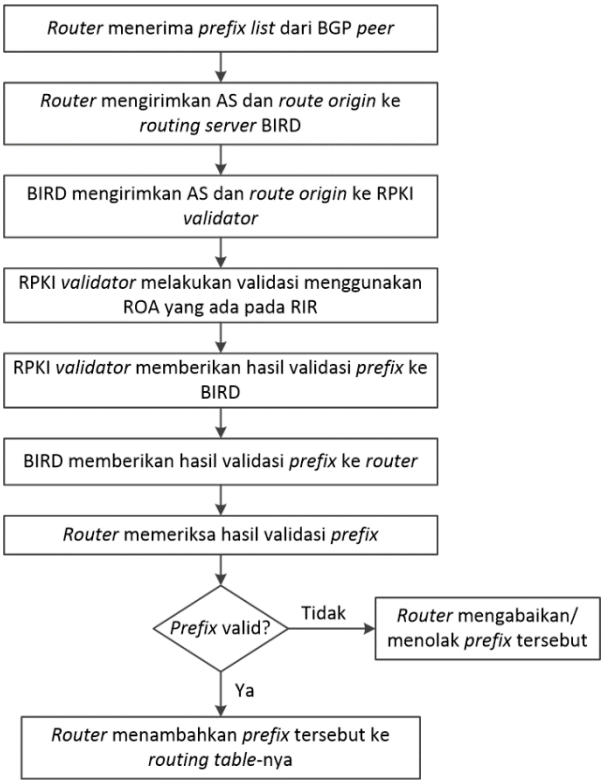


Gambar 1. Tahapan Penelitian

Berikut penjelasan Gambar 1. Proses dimulai dengan mengidentifikasi masalah, yaitu bagaimana mengimplementasikan protokol BGP dan RPKI menggunakan BIRD ke dalam suatu AS. Tahap kedua yaitu melakukan analisis topologi dan kebutuhan sistem dalam proses simulasi. Ketiga adalah melakukan perancangan topologi serta implementasi protokol BGP dan RPKI menggunakan BIRD ke dalam simulasi menggunakan GNS3. Tahap keempat yaitu melakukan pengujian *filtering prefix* menggunakan RPKI. Tahap terakhir adalah penyimpulan hasil. Setelah implementasi dan pengujian selesai, dilakukan penyimpulan hasil dari implementasi.

3. Hasil dan Pembahasan

Hasil dari penelitian ini adalah implementasi BIRD yang bekerja pada sistem operasi *Ubuntu* sebagai *route reflector* dan penerapan protokol BGP untuk proses *routing* antar AS, serta RPKI untuk melakukan validasi setiap *prefix* dengan AS-nya. Proses validasi *prefix* atau *subnet* dilakukan menggunakan aplikasi *validator* yaitu *Routinator* [22]. Proses validasi ini ditunjukkan pada Gambar 2.



Gambar 2. Diagram alir proses validasi dengan RPKI

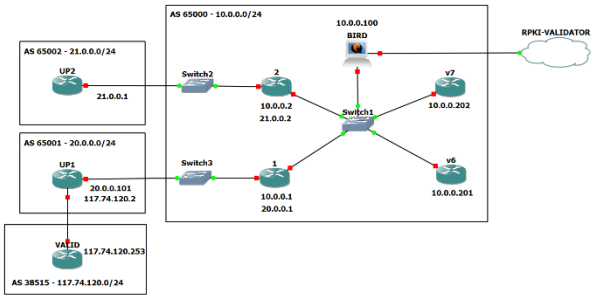
Router menerima *prefix* dari BGP *peer* dan meneruskan AS dan *route origin* ke *routing server* BIRD menggunakan protokol BGP. *Routing server* BIRD kemudian meneruskan data yang diterima ke RPKI *validator* untuk diperiksa. RPKI *validator* kemudian melakukan validasi data yang diterima dengan ROA yang ada pada RIR yang berperan sebagai CA. Dalam proses ini, RPKI *validator* menggunakan kunci publik yang diterbitkan oleh CA untuk melakukan validasi. Setelah proses validasi selesai, RPKI *validator* akan mengirimkan data hasil validasi ke *routing server* BIRD menggunakan protokol RTR, yang kemudian diteruskan (menggunakan protokol BGP) ke *router* yang bersangkutan agar dapat menentukan *prefix* mana yang dapat diterima untuk disimpan ke *routing table*-nya. Di sisi lain, AS wajib mendaftarkan nomor AS beserta *prefix* yang dimiliki ke RIR agar dapat ditambahkan ke dalam ROA. Setelah proses pendaftaran, AS akan menerima *letter of authorization* dari RIR yang berupa

DOI: https://doi.org/10.29207/resti.v5i6.3631

surat fisik untuk menyatakan bahwa nomor AS beserta *prefix* yang dimilikinya telah didaftarkan.

3.1. Perancangan Simulasi dan Topologi

Perancangan simulasi dan topologi dilakukan dengan menggunakan dua buah komputer, komputer pertama merupakan *server* RPKI *validator* dan komputer kedua menjalankan GNS3 yang mengimplementasikan *router* dan *routing server* BIRD secara virtual. GNS3 memanfaatkan *VMWare* untuk menjalankan *Virtual Machine* (VM) GNS3 yang berguna untuk menjalankan beberapa *router virtual* dan satu komputer yang bertugas sebagai *routing server* BIRD yang terhubung ke *server* RPKI *validator* melalui koneksi Internet. Topologi jaringan yang digunakan dalam simulasi ditunjukkan pada Gambar 3.



Gambar 3. Topologi pada GNS3

Router yang digunakan adalah *MikroTik* CHR v6.47 sebanyak enam unit dan *MikroTik* CHR v7.0b8 sebanyak satu unit. Komputer untuk *routing server* yang menjalankan BIRD menggunakan sistem operasi *Ubuntu* 18.04 LTS. Empat *router* (*router* 1, *router* 2, *router* v7, dan *router* v6) dan satu komputer *routing server* berada dalam satu *subnet* dan AS, yaitu AS 65000 dengan *subnet* 10.0.0.0/24, sedangkan dua *router* lainnya (*router* UP1 dan *router* UP2) disimulasikan sebagai *upstream router* dengan nomor AS 65001 dan AS 65002. AS 65001 mempunyai *subnet* 20.0.0.0/24 dan AS 65002 mempunyai 21.0.0.0/24. IP *address* dari *router* UP1 adalah 20.0.0.101 dan IP *address* dari *router* UP2 adalah 21.0.0.1.

Router UP1 dan *router* UP2 bertindak sebagai *upstream router* dan akan melakukan proses *advertise subnet* yang *invalid* dan *unknown*. Selain itu, terdapat 1 *router* yang bernama VALID dengan AS 38515 dan IP *address* 117.74.120.253 yang akan mengumumkan AS dan *subnet* yang *valid*, dan terhubung ke UP1. Daftar *subnet* yang di-*advertise* oleh *router* UP1, UP2 dan VALID ditunjukkan pada Tabel 2. Jumlah *subnet* yang di-advertise oleh router UP1 dan UP2 masing-masing adalah 14 *subnet* dan 12 *subnet*.

Komputer untuk *routing server* yang menjalankan BIRD terhubung melalui Internet dengan *server* yang menjalankan *Routinator* yang berperan sebagai RPKI *validator* untuk melakukan validasi *subnet*.

Mengacu pada topologi yang ditunjukkan pada Gambar 3, skenario yang dibuat dan akan diuji keberhasilan implementasinya dalam penelitian ini adalah sebagai berikut: Terdapat dua buah *router upstream* (UP1 dan UP2) yang akan melakukan proses *advertise subnet*. Dalam *advertisement* tersebut terdapat banyak subnet yang *invalid* atau *unknown*. Hasil yang diharapkan adalah agar *subnet* yang *invalid* dan *unknown* tidak akan masuk atau diterima sampai ke *router* v6 dan dan *router* v7. Sebelum penerapan RPKI, semua *subnet* yang di-*advertise* (*valid*, *invalid*, dan *unknown*) akan diterima seluruhnya sampai ke *router* v6 dan *router* v7. Setelah diterapkan RPKI, *router* 1 dan *router* 2 masih menerima *subnet* yang *invalid* dan *unknown*, tetapi sifatnya hanya menerima saja (karena *router* 1 terhubung secara langsung dengan *router* UP1 dan *router* 2 terhubung secara langsung dengan *router* UP2). Advertisement untuk subnet yang invalid dan *unknown* tidak akan diteruskan ke *router* lain yang ada dalam AS 65000 (*router* v6 dan *router* v7). *Router* 1 dan *router* 2 dalam AS 65000 masing-masing berperan sebagai BGP *peer* dari *router* UP1 pada AS 65001 dan *router* UP2 pada AS 65002.

Tabel 2. *Routing Advertisement*

No	*Router*	IP *address*	*Subnet* yang di-*advertise*
1	UP1	20.0.0.101	30.0.0.0/24
			40.0.0.0/24
			50.0.0.0/24
			60.0.0.0/24
			70.0.0.0/24
			80.0.0.0/24
			90.0.0.0/24
			91.0.0.0/24
			92.0.0.0/24
			93.0.0.0/24
			100.0.0.0/24
			101.0.0.0/24
			102.0.0.0/24
			103.0.0.0/24
2	UP2	21.0.0.1	130.0.0.0/24
			140.0.0.0/24
			150.0.0.0/24
			160.0.0.0/24
			170.0.0.0/24
			171.0.0.0/24
			172.0.0.0/24
			173.0.0.0/24
			180.0.0.0/24
			181.0.0.0/24
			182.0.0.0/24
			183.0.0.0/24
3	VALID	117.74.120.253	117.74.120.0/24

3.2. Instalasi BIRD dan RPKI *validator*

Instalasi BIRD dilakukan pada komputer yang bertugas sebagai *routing server*. Komputer yang dipakai berbasis *Ubuntu* 18.04 LTS dan berjalan secara *virtual* di atas GNS3 yang memanfaatkan VM GNS3. Setelah proses instalasi BIRD selesai, maka dapat dilakukan instalasi RPKI *validator*.

RPKI *validator* merupakan suatu aplikasi yang digunakan untuk proses validasi suatu *subnet* ketika

DOI: https://doi.org/10.29207/resti.v5i6.3631

proses *routing* dilakukan. Aplikasi yang digunakan adalah *Routinator* dan di-*install* pada komputer (*server*) yang terpisah dari komputer yang menjalankan VM GNS3. *Server* ini terkoneksi ke Internet untuk melakukan proses validasi. *Routinator* akan memperbarui *database* yang dimilikinya secara rutin dan mengambil data dari 5 RIR, yaitu APNIC, AFRINIC, ARIN, RIPE, dan LACNIC.

Setelah proses instalasi selesai, *Routinator* dapat dikonfigurasi dengan perintah berikut.

Konfigurasi *Routinator*

```
Routinator init --accept-arin-rpa
Routinator --verbose server -rtr
117.74.122.5:3323 –http 117.74.122.5:8323
```

Tujuan dari perintah di atas adalah untuk membangun *database* yang digunakan pada proses validasi. IP *address* dari *Routinator* adalah 117.74.122.5 dan menggunakan *port* 3323 untuk *validator* dan *port* 8323 untuk mengakses halaman menu *Routinator*. Halaman menu dapat diakses melalui *browser* dengan mengetikkan http://117.74.122.5:8323 pada *address bar* di *browser*.

3.3. Implementasi Protokol BGP dan RPKI

Setelah memastikan BIRD telah berjalan, maka dapat dilakukan implementasi protokol BGP. File untuk konfigurasi BIRD dapat ditemukan pada folder */etc/bird* kemudian pilih file *bird.conf* yang dapat diubah menggunakan *Text Editor*.

BIRD digunakan sebagai *route reflector* sehingga dapat menggunakan *template* telah tersedia yang dapat diakses sebagai acuan bagi *router* yang merupakan client dari BIRD. Sebelum mengimplementasikan protokol BGP dan RPKI, ada beberapa hal yang harus dikonfigurasikan terlebih dahulu, seperti *router id*, *log*, *ROA table*, *protocol device*, *protocol direct*, *protocol kernel* dan *protocol bfd*. Keseluruhan konfigurasi BIRD adalah sebagai berikut.

Konfigurasi bird.conf

```
log syslog all;
log syslog { debug, trace, info, remote,
warning, error, auth, fatal, bug };
log stderr all;

router id 10.0.0.100;

roa4 table ROA4;
roa6 table ROA6;

protocol kernel {
    learn;
    persist;
    ipv4 {
      import all;
      export filter {
          if proto = "direct1" then reject;
          accept;
          };
        };
  }

protocol device {
    scan time 60;
  }

protocol direct {
    ipv4;
    interface "ens3";
  }

protocol bfd {
    interface "ens3" {
      interval 50 ms;
    };
  }

protocol rpki VALIDATOR {
    roa4 {table ROA4;};
    roa6 {table ROA6;};
    remote 117.74.122.5;
    port 3323;
    refresh keep 30;
    retry keep 30;
    expire keep 3600;
    transport tcp;
  }

function is_v4_rpki_invalid () {
    return roa_check(ROA4, net,
bgp_path.last_nonaggregated) ~ [ROA_INVALID,
ROA_UNKNOWN];
  }

function is_v6_rpki_invalid () {
    return roa_check(ROA6, net,
bgp_path.last_nonaggregated) ~ [ROA_INVALID,
ROA_UNKNOWN];
  }

protocol static {
    ipv4;
    route 20.0.0.0/24 via 10.0.0.1;
    route 21.0.0.0/24 via 10.0.0.2;
  }

template bgp rr_clients {
    local 10.0.0.100 as 65000;
    neighbor as 65000;
    bfd;
    multihop;
    rr client;
    rr cluster id 10.0.0.100;

    ipv4 {
      gateway recursive;
      import filter {
        if is_v4_rpki_invalid() then reject;
        if bgp_next_hop.mask(24) =
from.mask(24) then {
          gw = bgp_next_hop;
          accept;
        }
        gw = from;
        accept;
      };
      import keep filtered;
      export where source ~
[RTS_BGP,RTS_STATIC];
    };

    ipv6 {
      import all;
      export where source = RTS_BGP;
    };
    }

protocol bgp border1 from rr_clients {
    neighbor 10.0.0.1;
  }

protocol bgp border2 from rr_clients {
    neighbor 10.0.0.2;
  }

protocol bgp client1 from rr_clients {
    neighbor 10.0.0.201;
```

DOI: https://doi.org/10.29207/resti.v5i6.3631

```
  }

protocol bgp client2 from rr_clients {
    neighbor 10.0.0.202;
}
```

Sebelum implementasi protokol BGP, dilakukan implementasi RPKI terlebih dahulu agar proses *filtering* dapat berjalan. Untuk implementasi RPKI, dibuat tabel ROA4 dan ROA6 yang akan digunakan pada proses *filtering*. Konfigurasi untuk pembuatan tabel dilakukan sebagai berikut.

Konfigurasi tabel ROA4 dan ROA6 pada bird.conf

```
  roa4 table ROA4;
  roa6 table ROA6;
```

Setelah tabel dibuat, dilakukan konfigurasi RPKI beserta fungsi-fungsi tambahannya. Konfigurasi RPKI pada BIRD dilakukan sebagai berikut.

Konfigurasi RPKI pada bird.conf

```
  protocol rpki VALIDATOR {
    roa4 {table ROA4;};
    roa6 {table ROA6;};
    remote 117.74.122.5;
    port 3323;
    refresh keep 30;
    retry keep 30;
    expire keep 3600;
    transport tcp;
  }
```

Baris *roa4* dan *roa6* digunakan agar *Routinator* dapat mengakses tabel ROA4 dan ROA6 yang dibuat sebelumnya. Selanjutnya, IP *address* beserta *port number* yang digunakan oleh *Routinator* diisikan pada baris *remote* dan *port* untuk menghubungkan BIRD dengan *Routinator* agar proses validasi dapat dilakukan. Baris sisanya digunakan untuk interval pembaruan (*refresh keep*), waktu *expire* (*expire keep*), serta protokol *transport* yang digunakan (*transport*).

Selain konfigurasi RPKI di atas, diperlukan fungsi-fungsi tambahan yang akan digunakan untuk melakukan proses *filtering*. Fungsi-fungsi ini ditambahkan tepat setelah konfigurasi RPKI karena fungsi tersebut akan melakukan proses validasi melalui RPKI *validator* terlebih dahulu sebelum melakukan *filtering*. Terdapat dua buah fungsi: 1) untuk *filtering subnet IPv4* dan 2) untuk *filtering subnet IPv6.* Fungsi tersebut akan mencocokkan *subnet* yang di-*advertise* dengan data yang diperoleh dari ROA *table* yang dimiliki oleh RPKI. Berikut adalah fungsi yang ditambahkan ke dalam file *bird.conf.*

Konfigurasi fungsi pada bird.conf

```
  function is_v4_rpki_invalid () {
    return roa_check(ROA4, net,
bgp_path.last_nonaggregated) ~ [ROA_INVALID,
ROA_UNKNOWN];
  }

  function is_v6_rpki_invalid () {
    return roa_check(ROA6, net,
bgp_path.last_nonaggregated) ~ [ROA_INVALID,
ROA_UNKNOWN];
  }
```

Dengan konfigurasi tersebut, maka setiap *subnet* yang *invalid* maupun *unknown* akan langsung ditolak. Hanya *subnet* yang *valid* yang dapat diterima.

Konfigurasi untuk protokol BGP menggunakan metode *route reflector* sehingga tidak perlu menuliskan *setting* yang sama untuk setiap *router* yang akan menjadi *client* dari BIRD. Pengaturan pada setiap *router* mengikuti *template* yang telah tersedia. Berikut adalah kutipan dari file *bird.conf* untuk konfigurasi dasar *template*.

Konfigurasi dasar template BGP pada bird.conf

```
  template bgp rr_clients {
    local 10.0.0.100 as 65000;
    neighbor as 65000;
    bfd;
    multihop;
    rr client;
    rr cluster id 10.0.0.100;
```

Setiap *router* yang menjadi *client* mengambil konfigurasi BGP pada bagian *template bgp rr_clients* yang telah dibuat. Bagian *local* harus diisikan sesuai dengan nomor AS dan IP *address* dari komputer yang menjalankan BIRD, kemudian *neighbor* juga diisikan dengan AS yang sama dengan bagian *local*. Baris *bfd* digunakan agar BIRD dapat menerima pesan jika terjadi kegagalan antara dua perangkat dengan memanfaatkan protokol *Bidirectional Forwarding Detection* (BFD). Baris *multihop* digunakan agar konfigurasi BGP dapat disampaikan ke *neighbor* yang tidak terhubung secara langsung. Baris *rr client* dan *rr cluster* digunakan agar BIRD dapat berperan sebagai *route reflector* dan tiap *neighbor* berlaku sebagai *client*. Baris *rr cluster* diisi dengan *router id* agar tiap *router* yang menjadi *client* dapat dijadikan satu *cluster* dengan *route reflector*.

Proses *filtering* diimplementasikan pada bagian *ipv4* dalam file *bird.conf*. *Filtering* dalam hal ini menggunakan *protocol rpki VALIDATOR* yang telah dikonfigurasi sebelumnya. Pada konfigurasi bagian *ipv4* terdapat perintah yang akan memanggil fungsi tambahan pada RPKI untuk menolak *subnet* yang *unknown* dan *invalid* pada ROA sehingga tidak dimasukkan ke dalam *routing table* oleh *router*. Penelitian ini hanya menggunakan IPv4 sehingga pada bagian *ipv6* tidak diubah konfigurasinya. Berikut adalah kutipan dari *bird.conf* untuk pengaturan proses *filtering* pada BGP menggunakan RPKI.

Konfigurasi filtering BGP pada bird.conf

```
  ipv4 {
      gateway recursive;
      import filter {
        if is_v4_rpki_invalid() then reject;
        if bgp_next_hop.mask(24) =
from.mask(24) then {
          gw = bgp_next_hop;
          accept;
        }
        gw = from;
        accept;
      };
      import keep filtered;
      export where source ~
[RTS_BGP,RTS_STATIC];
```

DOI: https://doi.org/10.29207/resti.v5i6.3631

```
  };

  ipv6 {
    import all;
    export where source = RTS_BGP;
  };
```

Bagian yang terakhir adalah membuat konfigurasi agar tiap *router* yang terhubung menjadi *client* dari BIRD. Setiap *router* yang akan dijadikan *client* dibuatkan konfigurasi masing-masing dan harus diisikan dengan IP *address* dari *router* tersebut. Berikut adalah konfigurasi untuk *route reflector*.

Konfigurasi route reflector client pada bird.conf

```
protocol bgp border1 from rr_clients {
  neighbor 10.0.0.1;
}

protocol bgp border2 from rr_clients {
  neighbor 10.0.0.2;
}

protocol bgp client1 from rr_clients {
  neighbor 10.0.0.201;
}

protocol bgp client2 from rr_clients {
  neighbor 10.0.0.202;
}
```

Perhatikan bahwa 10.0.0.1, 10.0.0.2, 10.0.0.201, dan 10.0.0.202 masing-masing adalah alamat pada *Router* 1, *Router* 2, *Router* v6, dan *Router* v7. Dengan demikian, keempat *router* ini adalah *client* dari BIRD.

3.4. Hasil Simulasi

Mengacu pada topologi yang ditunjukkan pada Gambar 3 dan skenario yang dipaparkan pada bagian 3.1, hasil yang diperoleh dari simulasi sebelum penerapan RPKI ditunjukkan pada Tabel 3(a) yang merupakan *routing table* pada *router* v7. Dapat dilihat bahwa *subnet* yang di-*advertise* oleh *router* UP1 ada pada *routing table*. Hal ini berarti subnet tersebut diterima oleh *router* v7. Hasil ini dapat dibandingkan dengan Tabel 3(b) yang menunjukkan *routing table* pada *router* v7 setelah penerapan RPKI. Terlihat bahwa hanya *subnet* 117.74.120.0/24 yang di-*advertise* oleh *router* VALID pada AS 38515 yang diterima oleh *router* v7.

Tabel 3. *Routing Table* pada *Router* v7 sebelum dan sesudah penerapan RPKI

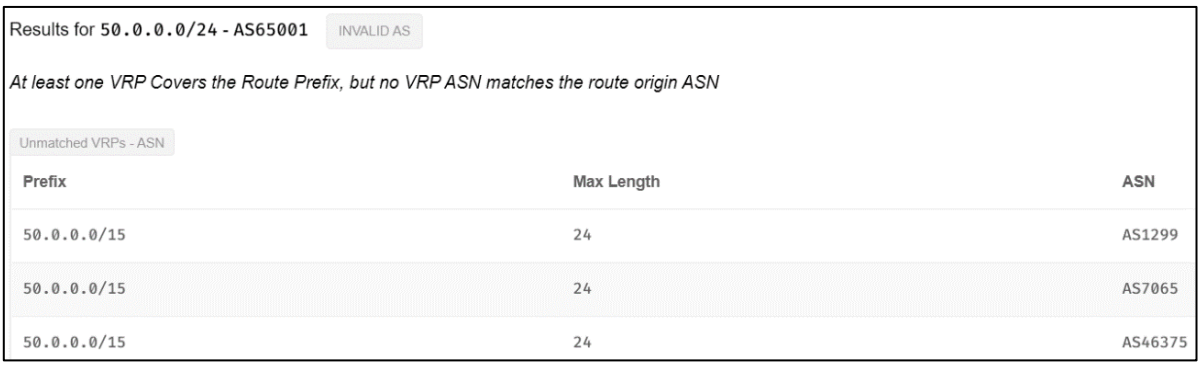
(a) Sebelum

```
[admin@MikroTik] > ip route print
Flags: D - DYNAMIC; I - INACTIVE, A - ACTIVE;
Columns: DST-ADDRESS, GATEWAY, DISTANCE
       DST-ADDRESS      GATEWAY     DIS
  DAC  10.0.0.0/24      ether1        0
  DAb  20.0.0.0/24      10.0.0.1    200
  DAb  21.0.0.0/24      10.0.0.2    200
  DIb  30.0.0.0/24      20.0.0.101  200
  DIb  40.0.0.0/24      20.0.0.101  200
  DIb  50.0.0.0/24      20.0.0.101  200
  DIb  60.0.0.0/24      20.0.0.101  200
  DIb  70.0.0.0/24      20.0.0.101  200
  DIb  80.0.0.0/24      20.0.0.101  200
  DIb  90.0.0.0/24      20.0.0.101  200
  DIb  91.0.0.0/24      20.0.0.101  200
  DIb  92.0.0.0/24      20.0.0.101  200
  DIb  93.0.0.0/24      20.0.0.101  200
  DIb  100.0.0.0/24     20.0.0.101  200
  DIb  101.0.0.0/24     20.0.0.101  200
  DIb  102.0.0.0/24     20.0.0.101  200
  DIb  103.0.0.0/24     20.0.0.101  200
  DIb  117.74.120.0/24  20.0.0.101  200
  DIb  130.0.0.0/24     21.0.0.1    200
  DIb  140.0.0.0/24     21.0.0.1    200
  DIb  150.0.0.0/24     21.0.0.1    200
  DIb  160.0.0.0/24     21.0.0.1    200
  DIb  170.0.0.0/24     21.0.0.1    200
  DIb  171.0.0.0/24     21.0.0.1    200
  DIb  172.0.0.0/24     21.0.0.1    200
  DIb  173.0.0.0/24     21.0.0.1    200
  DIb  180.0.0.0/24     21.0.0.1    200
  DIb  181.0.0.0/24     21.0.0.1    200
  DIb  182.0.0.0/24     21.0.0.1    200
  DIb  183.0.0.0/24     21.0.0.1    200
```

(b) Sesudah

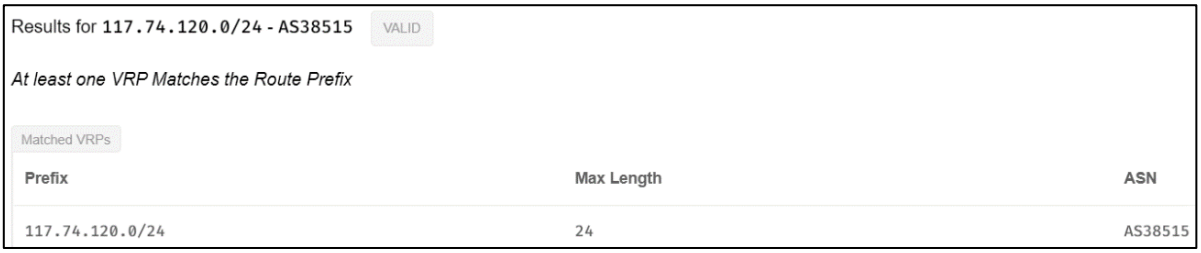
```
[admin@MikroTik] > ip route print
Flags: D - DYNAMIC; I - INACTIVE, A - ACTIVE;
Columns: DST-ADDRESS, GATEWAY, DISTANCE
       DST-ADDRESS      GATEWAY     DIS
  DAC  10.0.0.0/24      ether1        0
  DAb  20.0.0.0/24      10.0.0.1    200
  DAb  21.0.0.0/24      10.0.0.2    200
  DIb  117.74.120.0/24  20.0.0.101  200
```

DOI: https://doi.org/10.29207/resti.v5i6.3631



Gambar 4. Hasil validasi *prefix* pada *Routinator*: *Invalid Subnet*

Gambar 5. Hasil validasi *prefix* pada *Routinator*: *Unknown Subnet*



Gambar 6. Hasil validasi *prefix* pada *Routinator*: *Valid*

Selanjutnya, dapat diperiksa melalui *Routinator* apakah *subnet* tersebut benar *valid*, *invalid*, atau *unknown*. Hasil pemeriksaan melalui *Routinator* ditunjukkan pada Gambar 4, 5, dan 6. Gambar 4 menunjukkan pencarian untuk AS 65001 dengan *subnet* 50.0.0.0/24 di mana hasilnya adalah *invalid*. Gambar 5 menunjukkan pencarian *subnet* 183.0.0.0/24 yang di-*advertise* oleh *router* UP2 (AS 65002) dengan hasil *unknown*, yang berarti *subnet* tersebut tidak ada pada ROA yang divalidasi untuk AS 65002. VRP merupakan singkatan dari *validated* ROA *payload*. Gambar 6 menunjukkan pencarian *subnet* 117.74.120.0/24 dengan hasil *valid* pada AS 38515.

Pada sisi lain, setelah penerapan RPKI, *routing table* pada *router* 1 (dan *router* 2) tetap menerima *subnet* yang *invalid* dan *unknown* karena terhubung secara langsung dengan *router* UP1 (dan *router* UP2). Akan tetapi, router 1 dan router 2 tidak akan meneruskan *subnet* tersebut ke *router* lainnya dalam AS 65000. Tabel 4(a) menunjukkan *routing table* pada *router* 1 sebelum penerapan RPKI dan *routing table* setelah penerapan RPKI ditunjukkan pada Tabel 4(b). Terlihat bahwa *routing table* pada *router* 1 tidak lagi terdapat *subnet* (*invalid* dan *unknown*) yang di-*advertise* oleh router UP2 karena *router* 2 tidak meneruskan *advertisement* tersebut ke dalam AS 65000. Demikian juga *routing table* pada *router* 2 tidak memuat *subnet* (*invalid* dan *unknown*) yang di-*advertise* oleh *router* UP1 karena *router* 1 tidak meneruskan *advertisement* tersebut ke *router-router* lain dalam AS 65000.

4. Kesimpulan

Penelitian ini membahas implementasi protokol BGP dan RPKI dengan menggunakan BIRD untuk mengamankan suatu AS dari prefix *hijacking* atau BGP *hijacking*. BIRD berperan sebagai *route reflector* sehingga tidak perlu dilakukan konfigurasi pada tiap *router* yang menjadi *client* dari BIRD. Proses validasi *subnet* yang di-*advertise* oleh BGP *peer* dilakukan dengan cara menerapkan konfigurasi pada BIRD. BIRD kemudian akan meneruskan *subnet* yang diterima ke *Routinator* yang berperan sebagai RPKI *validator*. Pengujian sistem menunjukkan bahwa tiap *subnet* yang *invalid* dan tidak diketahui (*unknown*) akan ditolak dan tidak akan masuk ke dalam AS yang memanfaatkan. Dengan demikian, implementasi protokol BGP dan RPKI dengan BIRD berhasil mengamankan AS dari resiko BGP *hijacking* dengan hanya mengijinkan *subnet* yang *valid* yang dapat ditambahkan ke *routing table* pada router di dalam suatu AS.

Ucapan Terimakasih

Jurnal ini saya dedikasikan sebagai ucapan terima kasih kepada PT. Grahamedia Informasi, Salatiga yang telah memfasilitasi penelitian ini dengan menyediakan *server* untuk menjalankan *Routinator* serta mengijinkan penggunaan AS *number* beserta *subnet* yang *valid*.

DOI: https://doi.org/10.29207/resti.v5i6.3631

Tabel 4. *Routing Table* pada *Router* 1 sebelum dan sesudah penerapan RPKI

(a) Sebelum

```
[admin@MikroTik] > ip route print
Flags: X - disabled, A - active, D - dynamic, C - connect, S - static,
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 0 ADC  10.0.0.0/24        10.0.0.1        ether1                    0
 1 ADC  20.0.0.0/24        20.0.0.1        ether2                    0
 2  Db  20.0.0.0/24                        20.0.0.101               20
 3 A S  21.0.0.0/24                        10.0.0.2                  1
 4  Db  21.0.0.0/24                        10.0.0.2                200
 5 ADb  30.0.0.0/24                        20.0.0.101               20
 6 ADb  40.0.0.0/24                        20.0.0.101               20
 7 ADb  50.0.0.0/24                        20.0.0.101               20
 8 ADb  60.0.0.0/24                        20.0.0.101               20
 9 ADb  70.0.0.0/24                        20.0.0.101               20
10 ADb  80.0.0.0/24                        20.0.0.101               20
11 ADb  90.0.0.0/24                        20.0.0.101               20
12 ADb  91.0.0.0/24                        20.0.0.101               20
13 ADb  92.0.0.0/24                        20.0.0.101               20
14 ADb  93.0.0.0/24                        20.0.0.101               20
15 ADb  100.0.0.0/24                       20.0.0.101               20
16 ADb  101.0.0.0/24                       20.0.0.101               20
17 ADb  102.0.0.0/24                       20.0.0.101               20
18 ADb  103.0.0.0/24                       20.0.0.101               20
19 ADb  117.74.120.0/24                    20.0.0.101               20
20 ADb  130.0.0.0/24                       21.0.0.1                200
21 ADb  140.0.0.0/24                       21.0.0.1                200
22 ADb  150.0.0.0/24                       21.0.0.1                200
23 ADb  160.0.0.0/24                       21.0.0.1                200
24 ADb  170.0.0.0/24                       21.0.0.1                200
25 ADb  171.0.0.0/24                       21.0.0.1                200
26 ADb  172.0.0.0/24                       21.0.0.1                200
27 ADb  173.0.0.0/24                       21.0.0.1                200
28 ADb  180.0.0.0/24                       21.0.0.1                200
29 ADb  181.0.0.0/24                       21.0.0.1                200
30 ADb  182.0.0.0/24                       21.0.0.1                200
31 ADb  183.0.0.0/24                       21.0.0.1                200
```

(b) Sesudah

```
[admin@MikroTik] > ip route print
Flags: X - disabled, A - active, D - dynamic, C - connect, S - static,
 #      DST-ADDRESS        PREF-SRC        GATEWAY            DISTANCE
 0 ADC  10.0.0.0/24        10.0.0.1        ether1                    0
 1 ADC  20.0.0.0/24        20.0.0.1        ether2                    0
 2  Db  20.0.0.0/24                        20.0.0.101               20
 3 A S  21.0.0.0/24                        10.0.0.2                  1
 4  Db  21.0.0.0/24                        10.0.0.2                200
 5 ADb  30.0.0.0/24                        20.0.0.101               20
 6 ADb  40.0.0.0/24                        20.0.0.101               20
 7 ADb  50.0.0.0/24                        20.0.0.101               20
 8 ADb  60.0.0.0/24                        20.0.0.101               20
 9 ADb  70.0.0.0/24                        20.0.0.101               20
10 ADb  80.0.0.0/24                        20.0.0.101               20
11 ADb  90.0.0.0/24                        20.0.0.101               20
12 ADb  91.0.0.0/24                        20.0.0.101               20
13 ADb  92.0.0.0/24                        20.0.0.101               20
14 ADb  93.0.0.0/24                        20.0.0.101               20
15 ADb  100.0.0.0/24                       20.0.0.101               20
16 ADb  101.0.0.0/24                       20.0.0.101               20
17 ADb  102.0.0.0/24                       20.0.0.101               20
18 ADb  103.0.0.0/24                       20.0.0.101               20
19 ADb  117.74.120.0/24                    20.0.0.101               20
```

Daftar Rujukan

[1] Rekhter Y.; Li T. and Hares S., "A Border Gateway Protocol 4 (BGP-4)," 2006.

[2] D. Walton, A. Retana, E. Chen, and J. Scudder, "Advertisement of Multiple Paths in BGP - RFC 7911," *Internet Eng. Task Force*, pp. 1–8, 2016.

[3] "Quagga Software Routing Suite." Accessed: Nov. 08, 2021. [Online]. Available: https://www.quagga.net/.

[4] "The BIRD Internet Routing Daemon Project." Accessed: Sep. 29, 2021. [Online]. Available: https://bird.network.cz/.

[5] O. Filip, M. Mareš, O. Zajíček, and J. Matějka, "BIRD Internet Routing Daemon," [Online]. Available: http://labs.nic.cz.

[6] M. Lepinski and S. Kent, "An Infrastructure to Support Secure Internet Routing," Feb. 2012, doi: 10.17487/RFC6480.

[7] ICANN, "Resource Public Key Infrastructure Technical Analysis," 2020.

[8] P. Sermpezis, V. Kotronis, A. Dainotti, and X. Dimitropoulos, "A survey among network operators on BGP prefix hijacking," *Comput. Commun. Rev.*, vol. 48, no. 1, pp. 64–69, 2018, doi: 10.1145/3211852.3211862.

[9] "Is BGP safe yet? ·Cloudflare." Accessed: Sep. 28, 2021. [Online]. Available: https://isbgpsafeyet.com/.

[10] T. Chung *et al.*, "RPKI is coming of age: A longitudinal study of RPKI deployment and invalid route origins," *Proc. ACM SIGCOMM Internet Meas. Conf. IMC*, pp. 406–419, 2019, doi: 10.1145/3355369.3355596.

[11] M. Wählisch, O. Maennel, and T. C. Schmidt, "Towards detecting BGP route hijacking using the RPKI," *Comput. Commun. Rev.*, vol. 42, no. 4, pp. 103–104, 2012, doi: 10.1145/2377677.2377702.

[12] C. Lynn, S. Kent, and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers," Jun. 2004, doi: 10.17487/RFC3779.

[13] G. Huston and G. Michaelson, "Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs)," 2012. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6483.

[14] R. Bush and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol," Jan. 2013, doi: 10.17487/RFC6810.

[15] "GNS3 Network Simulator." https://www.gns3.com/.

[16] H. A. Musril, "Simulasi Interkoneksi Antara Autonomous System (As) Menggunakan Border Gateway Protocol (Bgp)," *InfoTekJar (Jurnal Nas. Inform. dan Teknol. Jaringan)*, vol. 2, no. 1, pp. 1–9, 2017, doi: 10.30743/infotekjar.v2i1.151.

[17] A. Fathurohman, "Border Gateway Protocol (BGP) Protocol Implementation in Public Network of the Universitas Muhammadiyah Semarang," vol. 2, no. 1, 2021.

[18] R. D. Marcus and E. Tfuakani, "Perancangan Jaringan Skala Besar dengan Menggunakan Metode Border Gateway Protocol (BGP) Berbasis Mikrotik," *Briliant J. Ris. dan Konseptual*, vol. 4, no. 3, p. 401, 2019, doi: 10.28926/briliant.v4i3.361.

[19] Y. Gilad, A. Cohen, A. Herzberg, M. Schapira, and H. Shulman, "Are We There Yet? On RPKI's Deployment and Security," no. February, 2017, doi: 10.14722/ndss.2017.23123.

[20] G. Chang, M. Arianezhad, and L. Trajkovic, "Using resource public key infrastructure for secure border gateway protocol," *Can. Conf. Electr. Comput. Eng.*, vol. 2016-Octob, 2016, doi: 10.1109/CCECE.2016.7726675.

[21] C. Testart, P. Richter, A. King, A. Dainotti, and D. Clark, "To Filter or Not to Filter: Measuring the Benefits of Registering in the RPKI Today," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 12048 LNCS, pp. 71–87, 2020, doi: 10.1007/978-3-030-44081-7_5.

[22] "NLnet Labs - RPKI Tools - Routinator." Accessed: Oct. 05, 2021. [Online]. Available: https://www.nlnetlabs.nl/projects/rpki/routinator/.

DOI: https://doi.org/10.29207/resti.v5i6.3631