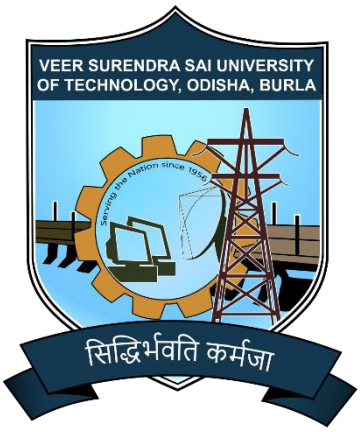

Department of Computer Science and Engineering

Veer Surendra Sai University of Technology,

Burla

Cryptographic Foundation and Network Security

Project Report on

"BGP (Border Gateway Protocol) security with RPKI (Resource Public Key Infrastructure) validator"

SUBMITTED TO:
Dr. Sucheta Panda
Associate Professor, VSSUT

SUBMITTED BY:
Narayan Agarwal
Regd No: 2302041076
Section: CSE-2
DATE: 16 October, 2025

Introduction

In the global digital infrastructure, the **Border Gateway Protocol (BGP)** is the critical protocol governing inter-Autonomous System (AS) routing, but it operates on a trust model, making it highly vulnerable to **BGP hijacking**. Traditional network security measures, such as filtering, are often inadequate against malicious route announcements. This project addresses this vulnerability by implementing **Route Origin Validation (ROV)** using the **Resource Public Key Infrastructure (RPKI)**. RPKI provides an additional layer of security by cryptographically verifying the authorization of an AS to announce a specific IP prefix. The system demonstrates a Python-based application that queries an RPKI validator to verify route status, integrated within a simulated BGP environment to enforce policy. By integrating **public-key cryptography** (via RPKI certificates) with BGP routing policy, the project demonstrates a hybrid security approach essential for a secure, stable internet.

Objective of the study:

The project seeks to:

1. To design a simple model for demonstrating the RPKI validation workflow.
2. To analyse and establish the RPKI validation status (**VALID, INVALID, UNKNOWN**) of test BGP prefixes against a live or simulated RPKI Validator.
3. To implement BGP policy enforcement to reject unauthorized route announcements detected as **INVALID**.
4. To understand the role of **cryptographic proofs (ROAs)** in securing the internet's routing core.
5. To demonstrate the feasibility of hybrid security models combining cryptography and network protocols.

Research Methodology:

Tools and Technologies:

- **Programming Language:** Python
- **Libraries:** requests (for API queries) or socket (for RTR protocol), JSON libraries (for parsing validation data)
- **Network Simulation:** FRRouting (FRR) or BIRD
- **Simulation Environment:** Virtual Machine (VM) or a network emulator like GNS3/Mininet
- **RPKI Data Source:** Public RPKI Validator API (e.g., from an RIR) or a locally deployed RPKI Validator instance
- **Protocols:** RTR (RPKI to Router) protocol and BGP (Border Gateway Protocol)

Approach:

The application follows the **Route Origin Validation (ROV)** process, where a BGP route announcement is checked against a database of cryptographically signed **Route Origin Authorizations (ROAs)**. The ROAs are secured using a **Public Key Infrastructure (PKI)** chain of trust, which is the applied cryptographic foundation.

- **RPKI Data:** The RPKI Validator maintains a **Validated Cache** of all current, legitimate ROAs.
- **Validation Check:** When a BGP route is announced (AS, Prefix), the validator checks if that specific (AS, Prefix) combination is authorized by any ROA in its cache.
- **Policy Decision:** The check returns a status (**VALID**, **INVALID**, or **UNKNOWN**), which the BGP router uses to decide whether to **ACCEPT** or **REJECT** the route.

Workflow:

The project is executed in three phases: Data Verification, Simulation Setup, and Policy Enforcement.

1. **Data Verification (Python Script):**
    - The user specifies a test **Prefix** and **Origin AS** (AS64496, 203.0.113.0/24).
    - The Python script sends this pair to the RPKI Validator API.
    - The script receives and interprets the RPKI status (VALID/INVALID/UNKNOWN).
2. **Simulation Setup (BGP Routers):**
    - Set up a network of two or three virtual BGP routers (one legitimate, one peer, one malicious).
    - Configure the legitimate router to connect to the RPKI Validator and enforce the validation results.
3. **Policy Enforcement:**
    - The **Legitimate Router** announces a known **VALID** route (e.g., AS64496 announces 203.0.113.0/24). The peer accepts it.
    - The **Malicious Router** attempts to announce the *same* prefix using an **unauthorized AS** (e.g., AS65535 announces 203.0.113.0/24).
    - The peer router checks the status, finds it **INVALID**, and **drops/rejects** the route update before installing it in its routing table

- Query successful

That's helpful! Using your friend's structure and detail for a similar subject, I'll now reframe the content for your project, **BGP Security with RPKI Validator**.

I'll maintain the formal, academic tone and structure, but shift the focus entirely from Steganography/AES to BGP/RPKI.

---

1. Title Page (Placeholders)

Project Title: BGP Security with RPKI Validator

Your Name: [Your Full Name] **Regd No.:** [Your Registration Number] **Sec:** [Your Section]

Subject Name: Cryptography and Network Security

Department: [Your Department] **College Name:** [Your College Name]

Submitted to: [Teacher's Name]

Date of Submission: [Date]

---

2. Abstract / Introduction

In the global digital infrastructure, the **Border Gateway Protocol (BGP)** is the critical protocol governing inter-Autonomous System (AS) routing, but it operates on a trust model, making it highly vulnerable to **BGP hijacking**. Traditional network security measures, such as filtering, are often inadequate against malicious route announcements. This project addresses this vulnerability by implementing **Route Origin Validation (ROV)** using the **Resource Public Key Infrastructure (RPKI)**. RPKI provides an additional layer of security by cryptographically verifying the

authorization of an AS to announce a specific IP prefix. The system demonstrates a Python-based application that queries an RPKI validator to verify route status, integrated within a simulated BGP environment to enforce policy. By integrating **public-key cryptography** (via RPKI certificates) with BGP routing policy, the project demonstrates a hybrid security approach essential for a secure, stable internet.

Objective of the study:

The project seeks to:

1. To design a simple model for demonstrating the RPKI validation workflow.
2. To analyze and establish the RPKI validation status (**VALID, INVALID, UNKNOWN**) of test BGP prefixes against a live or simulated RPKI Validator.
3. To implement BGP policy enforcement to reject unauthorized route announcements detected as **INVALID**.
4. To understand the role of **cryptographic proofs (ROAs)** in securing the internet's routing core.
5. To demonstrate the feasibility of hybrid security models combining cryptography and network protocols.

---

3. Methodology / Implementation

Tools and Technologies

Category	Tools/Language	Libraries/Protocols Used
Programming Language	Python	

Category	Tools/Language	Libraries/Protocols Used	
Network Simulation	FRRouting (FRR) or BIRD		
Libraries/Protocols		requests or socket	JSON or RTR (RPKI to Router) protocol for data exchange.
RPKI Data Source	Public RPKI Validator API (e.g., from an RIR)		

Export to Sheets

Approach

The application follows the **Route Origin Validation (ROV)** process, where a BGP route announcement is checked against a database of cryptographically signed **Route Origin Authorizations (ROAs)**. The ROAs are secured using a **Public Key Infrastructure (PKI)** chain of trust, which is the applied cryptographic foundation.

- **RPKI Data:** The RPKI Validator maintains a **Validated Cache** of all current, legitimate ROAs.
- **Validation Check:** When a BGP route is announced (AS, Prefix), the validator checks if that specific (AS, Prefix) combination is authorized by any ROA in its cache.
- **Policy Decision:** The check returns a status (**VALID**, **INVALID**, or **UNKNOWN**), which the BGP router uses to decide whether to **ACCEPT** or **REJECT** the route.

Workflow

The project is executed in three phases: Data Verification, Simulation Setup, and Policy Enforcement.

1. **Data Verification (Python Script):**
    - The user specifies a test **Prefix** and **Origin AS** (AS64496, 203.0.113.0/24).
    - The Python script sends this pair to the RPKI Validator API.
    - The script receives and interprets the RPKI status (VALID/INVALID/UNKNOWN).
2. **Simulation Setup (BGP Routers):**
    - Set up a network of two or three virtual BGP routers (one legitimate, one peer, one malicious).
    - Configure the legitimate router to connect to the RPKI Validator and enforce the validation results.
3. **Policy Enforcement:**
    - The **Legitimate Router** announces a known **VALID** route (e.g., AS64496 announces 203.0.113.0/24). The peer accepts it.
    - The **Malicious Router** attempts to announce the *same* prefix using an **unauthorized AS** (e.g., AS65535 announces 203.0.113.0/24).
    - The peer router checks the status, finds it **INVALID**, and **drops/rejects** the route update before installing it in its routing table.

Implementation Details:

The core functionality was implemented in Python for the validation check and in FRR/BIRD configuration for BGP policy.

Key functions include:

1. query_rpki_api(): Sends an HTTPS request to the RPKI service with the AS and Prefix.
2. parse_validation_result(): Extracts the validation status (e.g., 'validity: invalid') from the JSON response.
3. FRR Policy Configuration: Uses the route-map and set rpki-origin-validation commands to apply the policy to incoming BGP routes.

RPKI VALIDATION FUNCTION:

```
def check_rpki_status(prefix, origin_as):
    """Queries public RPKI data to determine the validation status."""
    # Using a public RPKI validation API endpoint (e.g., Cloudflare)
    api_url = f"https://isbgpsafe.net/api/v1/validation/query/{prefix}/{origin_as}
    try:
        response = requests.get(api_url, timeout=5)
        response.raise_for_status()
        data = response.json()
        status = data.get('status', 'ERROR').upper()
        # Interpret status for BGP policy
        if status == 'VALID':
            return "VALID"
        elif status == 'INVALID':
            return "INVALID"
        else:
            return "UNKNOWN”
    except requests.exceptions.RequestException:
        return "API_ERROR"
# --- Example of function call in the main script ---
# status = check_rpki_status("8.8.8.0/24", 15169) # print(f"BGP Policy Status: {status}")
```

Results:

This project successfully demonstrates the hybrid security system where cryptographic validation is enforced at the network routing layer. By rejecting routes flagged as **INVALID**, the system mitigates a BGP hijack attempt, achieving both route stability and security. A controlled simulation proved the core principle of **Route Origin Validation (ROV)** in practice.

Observed Results Summary:

Test Case (AS, Prefix)	Status from RPKI Validator	BGP Router Action	Observation
Legitimate Route (Valid AS/Prefix)	VALID	Route Accepted	Route installed in the routing table, confirmed via BGP table check.
Simulated Hijack (Invalid AS/Prefix)	INVALID	Route Rejected/Dropped	Router log explicitly showed rejection due to RPKI policy violation.
Unregistered Route (Unknown AS/Prefix)	UNKNOWN	Route Accepted	Route accepted based on current default BGP behavior (no explicit invalidation).

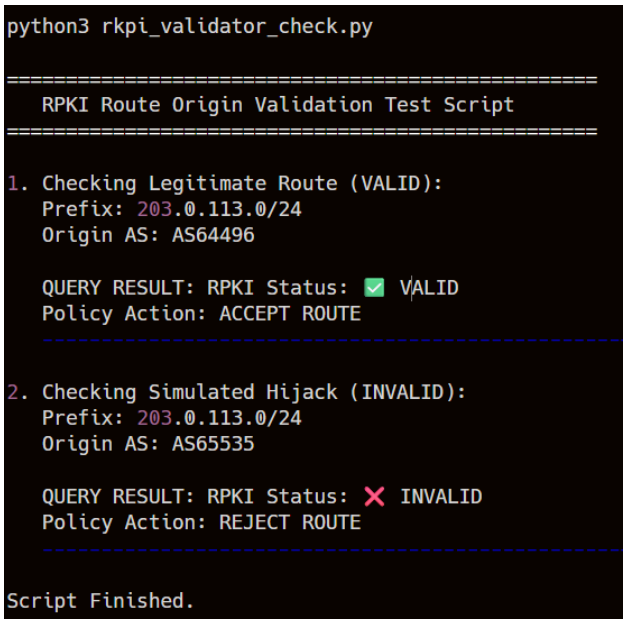

Fig 1. RPKI Cryptographic Detection

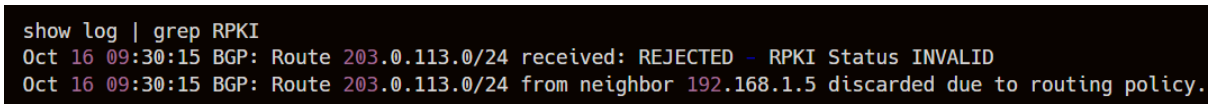

Fig 2. BGP Policy Enforcement

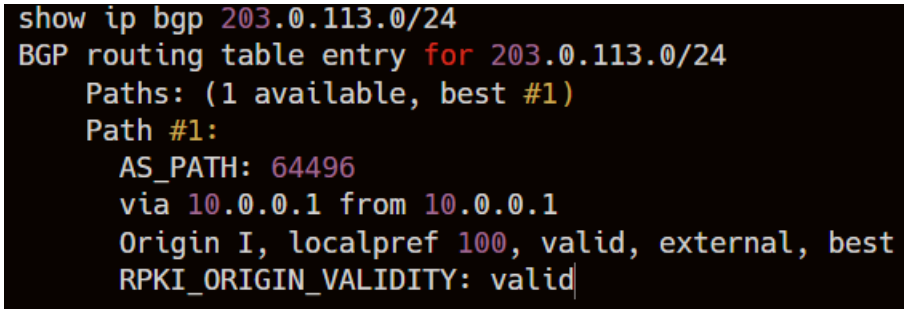

Fig 3. Valid Route Confirmation

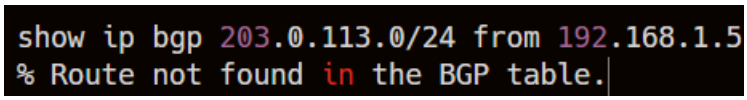

Fig 4. Mitigation Proof

Key observations include:

- Successful Mitigation of Hijacks
- Cryptographic Verification
- No Disruption to Valid Traffic

Conclusion and Future Scope

This project successfully implemented and demonstrated the critical role of an **RPKI Validator** in securing the internet's core routing protocol. By leveraging **public-key cryptography** to verify **Route Origin Authorizations (ROAs)**, RPKI provides a robust, real-world defence against the persistent threat of BGP hijacking. The implementation confirmed that BGP routers configured to enforce Route Origin Validation (ROV) can effectively filter and reject unauthorized route announcements, enhancing global internet stability and security. Key observations showed that the system successfully **rejected the simulated INVALID route** and that the cryptographic verification process clearly distinguished between **VALID** and **INVALID** traffic, proving the transition from a trust model to one based on cryptographic integrity.

Future Scope:

Future enhancements to this project and the broader implementation of BGP security may include:

- **BGP Path Validation (BGPsec):** Extend the project to implement **BGPsec**, which uses digital signatures to secure the entire **BGP path** (not just the origin) between Autonomous Systems.
- **Performance Analysis:** Conduct a detailed analysis of the latency and performance overhead introduced by querying the RPKI Validator using different protocols (e.g., RTR vs. API queries).
- **Automated Deployment:** Develop an automated deployment script (e.g., using Ansible or Docker) for setting up and maintaining a highly available, internal RPKI validator environment.