RPKI キャッシュサーバ運用における OS 等の環境依存性の分析とその緩和策に関する調査・研究

早田 優斗 [1,*1]　岡田 雅之 [1,*2]

概要：BGP には経路情報の正当性を検証する仕組みがなく，その対策として RPKI（Resource Public Key Infrastructure）を用いた ROV（Route Origin Validation）の導入が進められている．
RPKI における Validator は，リポジトリからリソース証明書や ROA（Route Origin Authorization）を取得・検証し，AS 番号と IP プレフィックスの正当な組み合わせを示す VRP（Validated ROA Payload）を生成する．
本研究では，OpenBSD 及び Ubuntu 上に Validator である rpki-client を導入し，VRP 件数や処理時間等の違いを実証的に分析し，OS 間の処理特性の差異を明確化した．
更に，NTP を無効化してシステム時刻を段階的に変更し，Validation の成否や取得件数の変動を詳細に検証した．
その結果，時刻のずれが大きいほど VRP 件数が減少し，Validation における時刻の正確性が信頼性に直結する要素であることが明らかとなった．
本研究は，RPKI 環境の信頼性確保に向けた設計及び運用指針の策定に資する．

キーワード：BGP，RPKI，OS，NTP

Analysis of OS and Environment Dependencies in RPKI Cache Server Operations and a Study on Their Mitigation

Yuto Soda[1,*]　Masayuki Okada[1,*2]

Abstract: BGP lacks a mechanism to validate the authenticity of routing information, and as a countermeasure, the introduction of RPKI (Resource Public Key Infrastructure)–based ROV (Route Origin Validation) has been progressing.
In RPKI, Validators retrieve and verify resource certificates and ROAs (Route Origin Authorizations) from repositories and generate VRPs (Validated ROA Payloads), which represent the legitimate combinations of AS numbers and IP prefixes.
In this study, we deployed the Validator rpki-client on both OpenBSD and Ubuntu, empirically analyzed differences in VRP counts and processing times, and clarified the processing characteristics across operating systems.
Furthermore, by disabling NTP and adjusting the system clock in stages, we conducted detailed validation of success rates and variations in VRP counts.
As a result, it was revealed that the larger the clock offset, the fewer VRPs were obtained, demonstrating that time accuracy directly affects the reliability of validation.
This research contributes to the formulation of design and operational guidelines for ensuring the reliability of RPKI environments.

Keywords: BGP, RPKI, OS, NTP

1. はじめに

1.1 背景と課題

インターネットの経路制御は BGP（Border Gateway Protocol）[1]により実現され，AS（Autonomous System）間で経路情報が交換されることでグローバルな到達性が維持されている．しかし，BGP そのものには経路広告の正当性（どの AS がどのプレフィックスを起源できるか）を検証する仕組みが備わっておらず，オペレーションミスや意図的な攻撃により，BGP ハイジャックやルートリーク[2]といったインシデントが発生し得るという本質的な課題がある．こうした事象は，トラフィックの誤誘導や通信断を引き起こし，上位サービスの信頼性に重大な影響を与えかねない．

この課題に対して，以前から広く運用されてきたのが IRR（Internet Routing Registry）に基づく経路フィルタである．運用者は RPSL（Routing Policy Specification Language）[3]に従って route, aut-num, as-set 等のオブジェクトを登録する．一方，対向となる事業者は WHOIS[4]等を通じてそれらを取得し，プレフィックスリストや AS パスフィルタを自動生成するという運用モデルが一般的である[3]．しかし，IRR は多数の独立したレジストリで構成され，運用ポリシーやデータ品質がレジストリごとに異なるうえ，資源保有者へ暗号学的に結び付けられた真正性の保証が前提とされてい

1 長崎県立大学 地域創生研究科 情報工学専攻 情報セキュリティコース
University of Nagasaki Graduate School of Regional Design and Creation Division of Computer Science Information Security course

*1 mc124002@sun.ac.jp
*2 okadams@sun.ac.jp

ないため，第三者による誤・不正登録，資源移転後の陳腐化したオブジェクトの残存，複数の IRR 間のミラー不整合といった恒常的な問題が指摘されてきた[5]. 実際, RIPE NCC は正当性が確認できない非権威（non-authoritative）な IRR オブジェクトを整理・廃止する方針（RIPE-731）を示しており，IRR における真正性が運用上の課題となり得ることが明示されている[6]. ARIN でも, 認証されていない ARIN-NONAUTH IRR を 2022 年に廃止するなど，非認証 IRR 情報への依存低減が進められている[7]．以上より，IRR ベースの経路フィルタは資源の正当な管理主体を暗号学的に証明・継承する仕組みが標準で備わっておらず，オブジェクトの真正性・完全性の担保に限界がある．結果として，IRR に基づく経路フィルタはセキュアとは言えず，BGP ハイジャックやルートリークの抑止に対して十分ではない．

以上の課題に対処するため，RFC 6480[8]で定義された RPKI（Resource Public Key Infrastructure）が整備されてきた．RPKI では, 該当の番号資源を保有する組織がリソース証明書（Resource Certificate）を取得し, この証明書に基づいて，IP アドレスと AS 番号の正当な組み合わせを示すデータである ROA（Route Origin Authorization）を発行する．運用面では，受信した経路情報の正当性を確認する処理である ROV（Route Origin Validation）を適用し，経路の Origin AS が ROA に記載された AS 番号と一致するかどうかを判定する．この ROV の処理方法は RFC 6811[9]にて明確化されており，国内においても JPNIC（日本ネットワークインフォメーションセンター）が ISP 等の組織向けに技術的ガイドラインを公開するなど，現場での導入・採用が進みつつある[10]．RPKI は，BGP の設計上の欠落を補完し得る基盤的なセキュリティ機構として位置づけられる．

RPKI， ROA， ROV の関係と典型的なフローを図 1 に示す（参考：[10]）．まず，レジストリ（JPNIC 等）から番号資源の割り当てを受けた組織は，当該資源を含むリソース証明書を取得し, その証明書に基づいて ROA を発行し，リポジトリへ rsync または RRDP[11]で公開し，格納する．Validator（ROA キャッシュサーバ）は，TAL（Trust Anchor Locator）[12]に基づき各リポジトリから ROA・Manifest[13]・CRL（Certificate Revocation List）[14]等を取得し，証明書・有効期間・完全性（Manifest とハッシュ照合）を検証したうえで VRP を生成し，validated cache に保存する．BGP ルータは，Validator から VRP（RTR[15]等で配布）を取得し，受信経路の Origin AS とプレフィックスを照合して ROV を実施し，RFC 6811 に従い Valid / Invalid / NotFound を判定する．これにより，誤った起源情報に基づく経路の受容を抑制できる．

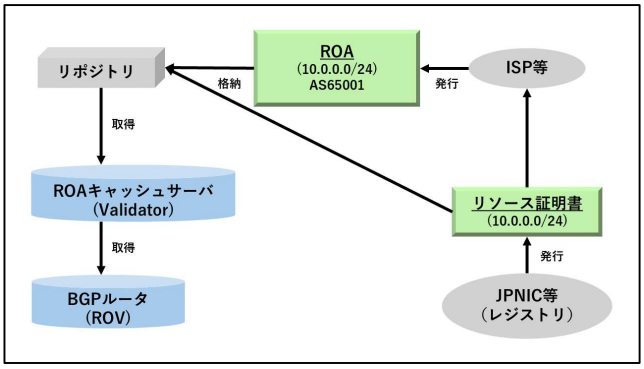

図 1 RPKI， ROA， ROV の仕組み及び関係性
Figure 1 RPKI, ROA, and ROV mechanisms and their relationships.

普及状況については，APNIC Labs の日本向けダッシュボード[16]で, アドレス空間ベースの ROA カバー率（Valid 比率）の時系列が公開されており，2025 年 8 月 22 日時点では IPv4 で 74.2%，IPv6 で 81.3%まで拡大していることが観察される．これは「どの程度のアドレス空間が ROA により起源正当化され得るか」を示す指標であり，実際のルータ側での ROV 導入率とは区別される点に留意が必要である．いずれにせよ ROA カバー率が一定水準に達した局面では，Validator の環境依存による VRP の欠落や判定の揺らぎを最小化することが，実運用の誤拒否・誤許可の回避と安定運用に直結する．

1.2 問題設定と研究目的

一方で，RPKI の実運用では，リポジトリからオブジェクト（ROA，証明書，Manifest，CRL 等）を取得・検証し，ルータが参照する VRP（Validated ROA Payload）を生成する役割を担う Validator の挙動が，全体の信頼性に直結する．学部時の先行検証では，rpki-client を含む Validator の実行環境（OS 等）により Validation 結果が相違する可能性を指摘した．しかし本研究で，NTP（Network Time Protocol）により両 OS の時刻を厳密に同期し，同一時刻で同時実行する条件を整えたところ，少なくとも当該条件下では Ubuntu と OpenBSD の rpki-client が導出する VRP Entries は一致することを確認した．この知見は，適切な運用条件（正確な時刻・同時実行・同一の取得方式）を満たせば，実装間で結果整合性が得られることを示す．一方で，処理時間（ユーザ時間・システム時間の内訳），取得失敗時の挙動，validated cache の更新・クリーンアップ件数，ログの可観測性などの運用特性には OS 依存の差異が存在し得ることを示唆する．

加えて, 本研究ではシステム時刻の同期状態が Validation 結果に与える影響を明確化する. RPKI の各オブジェクトには有効期間が定義されており，システム時刻が実際の時刻から乖離している場合, 検証の可否判定や VRP 生成に有意

な影響が現れる可能性がある．通常運用では NTP により時刻同期が行われるが，設定不備や一時的な障害等により同期が崩れる局面も想定し得る．

以上を踏まえ，本研究の目的は，(i) 同一条件下での rpki-client の結果整合性（VRP Entries の一致）を検証しつつ，（ii）OS 間で異なる運用特性（処理時間，エラーの表出，キャッシュの反映挙動等）を定量的に比較し，（iii）時刻同期の有無及びずれが Validation 結果に与える影響を実証的に評価し，現状のベストプラクティスとして有用な運用上の指針を提言することである．具体的には，OpenBSD 及び Ubuntu の両環境に同一バージョンの rpki-client を導入し，NTP で同期した同一時刻での同時実行の条件を整えたうえで挙動を比較する．更に，NTP を無効化してシステム時刻を 0 時間前（現時刻）から 24 時間前まで 1 時間刻みで段階的に変更し，VRP Entries・証明書パースエラー・処理時間等の変動特性を明らかにする．これらの分析から，OS 依存の運用特性の差と時刻同期の重要性を裏付け，実運用における検証環境の設計・評価・構築時に参照可能な指針を示すことを目指す．

2. 検証手法

2.1 前提条件

本研究は，rpki-client による Validation 結果に対して OS とシステム時刻が与える影響を明らかにすることを目的とし，以下の条件に基づいて比較検証を実施した．比較対象となる OS は，rpki-client が公式にサポートする OpenBSD 7.5 と，一般的な Linux ディストリビューションである Ubuntu 22.04 LTS の 2 種類である．両環境に同一バージョンの rpki-client 8.6 を導入し，各種条件下での実行ログ（標準出力を保存したもの）を収集・比較した．

比較にあたっては，各 OS で rpki-client が出力するログに含まれる統計情報（ROA 件数，VRP Entries，証明書パースエラー数等）を記録した．加えて，後段で示す結果比較に用いるため，ログに出力される処理時間の情報や validated cache に関する出力（例 : New files moved into validated cache の件数，Repository cleanup の内訳）も併せて保存した．これらは全て rpki-client の実行時に標準出力へ現れる情報に依拠し，外部計測ツールによる追加の計測は行っていない．

リポジトリの取得方式は，rpki-client の標準設定に従い RRDP を優先し，必要に応じて rsync にフォールバックした．なお，OpenBSD 環境では OpenRsync が，Ubuntu 環境では rsync がそれぞれ使用されるため，フォールバック発生時には内部実装の違いに起因して挙動差が生じる可能性がある．本稿では両実装が異なることを明記し，詳細な切り分けは考察で扱う．

2.2 検証 1

OS 間の比較に際しては，両環境とも NTP により現在時刻が正確に設定された状態で rpki-client を同一時刻に実行し，時刻のずれによる影響を排除したうえで，取得された VRP Entries，エラー出力，処理時間等を比較した．ここでの「同時」は，cron のスケジュール制御を用いて分単位で起動し，起動直後に sleep を用いて秒単位（30 秒）でそろえる操作を指す．これにより，リポジトリ側の更新タイミングの影響を排除している．

本研究では，Ubuntu 環境におけるシステム時刻の同期先を，デフォルトで設定されている”ntp.ubuntu.com”から，国立研究開発法人情報通信研究機構（NICT）の公開 NTP サーバである”ntp.nict.jp”に変更した．以下にその詳細な手順を示す．まず，「timedatectl status」を用いて初期状態を確認したところ，NTP サービスは無効（inactive）であり，システムクロックは同期していない状態であった．次に，「sudo timedatectl set-ntp true」を実行することで NTP サービスを有効化し，再度「timedatectl status」を確認した結果，NTP サービスが active となり，システムクロックが同期状態に移行したことを確認した．その時点で「timedatectl show-timesync --all」を実行したところ，”FallbackNTPServers=ntp.ubuntu.com”が設定されており，同期先は Ubuntu 既定の ”ntp.ubuntu.com”（ServerAddress=185.125.190.57，Stratum=2）であることが判明した．Stratum が 2 の場合，上流由来の遅延が累積し微小な時刻誤差が生じ得ることから[17], このままでは検証条件として望ましくないため，同期先を明示的に NICT に変更する必要があった．

そこで，NTP の設定ファイルである「/etc/systemd/timesyncd.conf」を編集した．初期状態では [Time] セクションに ”NTP=” が空欄状態で，”FallbackNTP=ntp.ubuntu.com”が記載されていた．これを”NTP=ntp.nict.jp”及び”FallbackNTP=”と変更した．すなわち，同期先を NICT の公開サーバである”ntp.nict.jp”に限定し，”FallbackNTP”を空欄とすることで Ubuntu 既定のサーバに切り替わらないようにした．その後，設定を反映させるために「sudo systemctl restart systemd-timesyncd」及び「sudo timedatectl set-ntp true」を実行し，サービスを再起動した．最後に，再度「timedatectl show-timesync --all」を実行し，設定が正しく反映されていることを確認した．その結果，”SystemNTPServers=ntp.nict.jp”，”ServerName=ntp.nict.jp”，”ServerAddress=133.243.238.243”，”Stratum=1”，”Reference=NICT”，”Jitter=0” と表示された．ここで，”SystemNTPServers”及び”ServerName”が”ntp.nict.jp”に変更されていること，”ServerAddress”に NICT の公開サーバ IP アドレス（例:133.243.238.243）が解決されていること，更に”Stratum=1”であることが確認できた．”Stratum=1”は直接原子時計や GPS 等の基準クロックと同期している上位階層

の時刻源であり, 時刻の精度が高いことを意味する[17][18]. Reference が"NICT"と表示されていることからも，確実にNICTの提供する基準クロックに同期していることが裏付けられる. 以上の手順により, Ubuntu 環境においてNTPサーバをNICTに統一した.

続いて, OpenBSD環境においても同様にNTP同期先をNICTに統一した．初期状態では”/etc/ntpd.conf”において”servers pool.ntp.org”及び”server time.cloudflare.com”が設定されており，実際に「ntpctl -s peers」を実行した結果からも, ”pool.ntp.org”や”time.cloudflare.com”が同期先として利用されていることが確認された. このままではUbuntuと同期先が一致せず，検証条件として望ましくないため，設定ファイルを編集し, 同期先をNICTに限定した. 具体的には，「/etc/ntpd.conf」を編集し，既存の”servers pool.ntp.org”及び”server time.cloudflare.com”をコメントアウトしたうえで，”server ntp.nict.jp”と記述した．また，”sensor *”の行は残して利用した. その後,「doas rcctl enable ntpd」及び「doas rcctl restart ntpd」を実行し，サービスを再起動した.

再起動後に「ntpctl -s peers」を実行したところ，同期先として”ntp.nict.jp”が表示され，”ServerAddress=61.205.120.130”, ”Stratum=1”であることが確認できた.

また，”offset=1.253ms”，”delay=13.412ms”，”jitter=0.129ms”といった測定値が得られ，時刻同期が十分に安定していることが示された. 以上により, OpenBSD環境においてもNTPサーバをNICTに統一し，Ubuntuと同一の基準クロックに同期させる条件が整った．これにより，OS間のrpki-client実行結果の比較において，NTPサーバ差が要因となる可能性を排除し，検証条件の妥当性を担保した.

同時実行の具体化にあたっては，Ubuntuではrootのcrontabにおいて「sudo crontab -e」を用い，事前にログ保存先ディレクトリを「sudo mkdir -p /home/siebold/rpki-client_cron_log」で作成したうえで，10分間隔・各回30秒後にrpki-clientを実行し，その都度標準出力を時刻付きの個別ファイルに保存する設定とした. 実際のエントリは「*/10 * * * * sleep 30 && /usr/local/sbin/rpki-client > /home/siebold/rpki-client_cron_log/rpki-client_$(date +%Y%m%d_%H%M%S).log 2>&1」である．OpenBSDでも同様にrootのcrontabを「doas crontab -e」で編集し，ログ保存先を「doas mkdir -p /home/siebold/rpki-client_cron_log」で準備したうえで，「*/10 * * * * sleep 30 && /usr/sbin/rpki-client > /home/siebold/rpki-client_cron_log/rpki-client_$(date +%Y%m%d_%H%M%S).log 2>&1」とした. いずれも起動は10分毎，sleepにより30秒後にrpki-clientが実行され，標準出力が同一のログファイルに保存される．これにより，各回の結果を相互に独立したファイルとして比較できるようにした.

2.3 検証 2

システム時刻の影響を検証するため，NTPを意図的に無効化し, システム時刻を過去に設定した状態でrpki-clientを実行した．設定時刻は，検証実行時点から0時間前（現時刻）から24時間前までを1時間刻みで段階的に変更し，各条件で得られたVRP Entries，証明書パースエラー，その他のログの有無を記録した.

本検証の自動化・再現性向上のため，シェルスクリプトを作成した. スクリプトは基準時刻をNTPで同期して取得したうえで, h=0...24（1時間刻み）について順次実行する．各回において，一時的にシステム時刻を切り替えつつ実行し，毎回の実行前に「現在基準時刻」「評価時刻（基準からh時間前）」をログの冒頭に明示する．Ubuntuでは，各回の開始時に「sudo timedatectl set-ntp true」でNTPを有効化し”NTPSynchronized=yes”であることを確認して基準時刻を取得する．h=0（0時間前）の回はそのままrpki-clientを実行する. h>0（1時間以上前）の回では, まず「sudo timedatectl set-ntp false」でNTPを無効化し，続いて「sudo timedatectl set-time 'YYYY-MM-DD HH:MM:SS'」で「基準時刻からh時間前」にシステム時刻を設定したのち，rpki-clientを実行する．実行直後に「sudo timedatectl set-ntp true」でNTPを再度有効化し，同期の復帰を確認してから次のhへ進む. OpenBSDにおいても同一方針で実施し，各回の開始時に「doas rcctl start ntpd」または起動済みであることを確認して基準時刻を取得する．h>0の回では「doas rcctl stop ntpd」でntpdを停止してから「doas date YYYYMMDDHHMM.SS」によりシステム時刻を「基準時刻からh時間前」に設定し，rpki-client実行後に「doas rcctl start ntpd」でNTP同期を復帰させる.

rpki-clientを実行する際は, 各回での干渉を避けるため独立のvalidated cacheを用いた. 例えば「/var/tmp/rpki-timeshift-cache/minus_hh」を都度新規に作成する．この際，rpki-clientが権限を降格する実行ユーザ（例:_rpki-client）が存在する環境では「sudo install -d -o _rpki-client -g _rpki-client -m 750 /var/tmp/rpki-timeshift-cache/minus_hh」（OpenBSDでは「doas install -d -o _rpki-client -g _rpki-client -m 750 …」）として所有者・権限を明示し，権限起因のスキップや書込み失敗を防止した. 実行コマンドは環境に応じて「/usr/local/sbin/rpki-client -j -d /var/tmp/rpki-timeshift-cache/minus_hh」または「/usr/sbin/rpki-client -j -d /var/tmp/rpki-timeshift-cache/minus_hh」とし，標準出力・標準エラーは回ごとの時刻付きログファイル（例:/home/siebold/rpki-client_timeshift_log/rpki-client_YYYYMMDD_HHMMSS_minus_hh.log）に保存，終了コードはログ末尾に記録した．各回は「NTP停止→時刻設定→rpki-client実行→NTP復帰」順で実施し，各回の終了後はNTP同期を必ず復帰させたうえで次回の基準時刻を再取得する手順とした．これにより，同期失敗の連鎖や各回の

キャッシュ汚染を抑制しつつ，評価時刻のみを変数とする条件を担保している．

ログは「/home/siebold/rpki-client_timeshift_log」に時刻付きファイル名で保存し，各回の標準出力・標準エラーを併せて記録した．rpki-client のキャッシュは回ごとに分離したディレクトリ（例：「/var/tmp/rpki-timeshift-cache/minus_xh」または「/var/cache/rpki-client」）を用い，rpki-client 内部の権限降格ユーザ（例：「_rpki-client」）に対して書込権限を付与することで，キャッシュ作成時の権限エラーを回避した．実行終了コード，処理時間，VRP Entries 等の統計指標は各ログ末尾に併記し，後述の集計・比較に利用した．

以上の手順により，OS の違い，時刻同期の有無，及び時刻のずれが rpki-client の Validation 結果に与える影響を，実行ログに基づいて記録・比較するための条件を整えた．なお，本章で述べた範囲を超える追加の実験・観測は行っておらず，以降の章では，ここで収集したログから得られた結果に基づいて考察を行う．

3. 検証結果と考察

3.1 検証 1

初めに，Ubuntu と OpenBSD 上で rpki-client を同時に実行した結果を比較した．両環境とも NTP により現在時刻が正確に設定されており，時刻のずれによる影響は排除されている．Ubuntu と OpenBSD で同一時刻かつ同一バージョンの rpki-client を同時実行したところ，VRP Entries は両 OS で完全に一致した（739,757 件，うち unique 733,415 件）．したがって，本検証条件では最終的な VRP Entries に OS 依存の不一致は認められなかった．これは，同一 TAL・同一リポジトリ群に対し，同一の検証規則を適用すれば同一の VRP Entries に収束するという整合性を実測で裏づけるものである．

一方，処理時間の内訳には OS 間での差異が顕著に現れた．Ubuntu は Processing time が 190 秒（user：183 秒，system：7 秒）であったのに対し，OpenBSD は Processing time が 376 秒（user：187 秒， system：192 秒）であった．総時間差の大半が system 時間に集中しており，Ubuntu の system 比率が約 3.7%に対し OpenBSD は約 51.1%と，カーネル空間での I/O やファイル操作に関する負荷で差が示唆される．ユーザ空間時間はほぼ同等であるため，rsync 実装差（Ubuntu の rsync / OpenBSD の OpenRsync）やファイルシステム・メタデータ処理の違いが主因候補と考えられるが，切り分けは今後の課題である．

validated cache の反映及びクリーンアップの統計にも差が見られた．Ubuntu では New files moved into validated cache が 327 件，removed directories が 278 件，Repository cleanup は kept 5714 件 / removed 8 件であった．OpenBSD ではそれぞれ 329 件，280 件，kept 5713 件 / removed 8 件であり，差はいずれも 2 件以内に収まっている．これらの差は取得順序やタイムスタンプ粒度，ディレクトリ走査順（readdir 順序差）等の非機能的要素に起因する微差である可能性が高く，最終的な VRP Entries への影響は確認されなかった．

ROA 件数は両 OS で 318,206 件（failed parse：29， invalid：0）と一致し，ASPA が 111 件，BGPsec Router Certificates が 2 件，TAL が 5 件，Manifests は総数 47,880 件で seqnum gaps が 1 件と，主要統計は同一であった．一方で，Certificates の non-functional は Ubuntu の 108 件に対し OpenBSD は 107 件，Manifests の failed parse は Ubuntu が 108 件に対し OpenBSD が 107 件，CRL 総数は Ubuntu が 47,772 件に対し OpenBSD が 47,773 件と，±1 件の差が散見された．いずれも「不成功側（failed / expired / non-functional）」のカウントであり，VRP 生成に寄与しない領域での誤差であるため，これらの微差が VRP Entries に波及しなかったという観測と整合する．

Ubuntu のログには，RRDP からの取得失敗による rsync フォールバック，rsync 接続エラー（socket IO code 10），DNS 解決失敗，TLS 読込エラー，rrdp notification の未更新によるスキップ，サーバ側の同時接続上限到達（“max connections reached”）等が混在して観測された．これらは rpki-client が新規取得に失敗した際に優先度順に手段を切り替えて処理を継続する回復動作が機能した結果であると解釈できる．すなわち，まず RRDP で notification.xml を参照し更新検知や delta/snapshot 取得を試行し，更新なしの場合や HTTP/TLS 層の障害があれば，rsync へフォールバックする．更に，rsync でも一時的な到達性低下や同時接続制限により取得が成立しない場合には，前回検証済みで有効期間内のオブジェクトを validated cache から再利用して VRP を再構成する．実験では最終的な VRP Entries が一致しており，これらの事象は回復動作が正常に機能した結果，瞬間的な取得失敗が VRP の欠落に直結しなかった事を示している．ただし，これは Manifest や CRL の有効期間に依存するため，障害が長期化した場合には VRP の減少や検証失敗に転じうる点には留意を要する．一方，OpenBSD のログでは本稿で提示した範囲では「CRL has expired」等の検証系メッセージが目立ち，ネットワーク系エラーの表出は少ない．これは，ログ表現の差（rsync / OpenRsync 起因のメッセージ差分）や，同一時刻でも到達経路・DNS 応答・IPv4 / IPv6 経路性の揺らぎが異なった可能性によるものと考えられる．もっとも，いずれの違いも最終的に VRP Entries には反映されていない．

以上より，同一時刻・同一設定下では VRP Entries は一致し，rpki-client の検証結果は OS に依存せず整合することを確認した．一方で，OS 間では (i) 処理時間，とりわけ system 時間の比率，(ii) validated cache の反映・クリーンアップ統計の微差，(iii) ログに現れるエラー種別・表現に違いが観測された．これらは，運用観点では障害時の復旧時間や監

視における閾値の設計，ログ相関分析に影響し得るため，SLA（Service Level Agreement）や手順書に反映すべき差異である．なお，本節の結果はあくまで当該時点の取得・到達状況に依存しており，rsync と OpenRsync の実装差，ファイルシステムや DNS / 経路性の影響，メタデータ解釈の違いなどの寄与度については追加切り分けが必要である．

3.2 検証 2

次に，評価時刻の影響を定量化するため，基準時刻（0 時間前）から 24 時間前までを 1 時間刻みで実行し，各回の VRP Entries を取得した．検証手法で前述の通り，各回の開始直前に NTP 同期で基準時刻を取得・記録したうえで，h=0 はそのまま実行する．h>0 では NTP を停止して「基準時刻から h 時間前」にシステム時刻を設定して rpki-client を実行し，直後に NTP を再度有効化して基準時刻へ復帰する手順とした（Ubuntu は「sudo timedatectl set-ntp true/false」「sudo timedatectl set-time 'YYYY-MM-DD HH:MM:SS'」，OpenBSD は「doas rcctl start/stop ntpd」「doas date YYYYMMDDHHMM.SS」を用いた）．各回では独立の検証キャッシュ（例:/var/tmp/rpki-timeshift-cache/minus_hh）を新規作成し，rpki-client が権限を降格する実行ユーザ（例:_rpki-client）に所有させて各回の干渉と権限起因の失敗を防止した．ログの冒頭には「基準時刻」「評価時刻」「使用キャッシュ」を明示し，末尾に終了コードと統計を出力した．表 1・表 2 に 1 時間毎の VRP Entries を，図 2・図 3 に時系列の推移を示す．

全体的な傾向として，両 OS とも時刻を過去へ遡るほど VRP Entries が減少し，最終的に 0 件へ収束するという共通の挙動が観測された．具体的には，0 時間前では Ubuntu が 740,846 件，OpenBSD が 740,845 件であり，1 件の差にとどまった．1 時間前は基準比で 9 割強（Ubuntu：680,751 件≒91.9%，OpenBSD：682,954 件≒92.2%）を維持する一方，2 時間前で約 3 分の 1（Ubuntu：262,598 件≒35.5%, OpenBSD：265,785 件≒35.9%）へ急減する転移点が共通して現れた．その後，11 時間前で基準比 1%未満（Ubuntu：6,024 件≒0.81%，OpenBSD：5,310 件≒0.72%）まで低下し，OpenBSD は 16 時間前，Ubuntu は 17 時間前で 0 件に到達した．5 時間前のみ一過的に差が拡大し（Ubuntu：56,001 件, OpenBSD：13,083 件），6 時間目前後で再び収束（Ubuntu：12,106 件，OpenBSD：12,122 件）した．これらの曲線形状は，ROA・Manifest・CRL に設定された有効期間及び検証要件に時刻が抵触した際，連鎖的に検証不能なオブジェクトが増加するという RPKI の設計と合致する．

0 時間前の 1 件差については，(i) 実行開始の秒単位の差に伴うリポジトリ側の境界更新（notBefore / nextUpdate 直近）との競合，(ii) 取得経路・実装差（Ubuntu では rsync，OpenBSD では OpenRsync）に伴う微小な反映順序の違い，(iii) 同一時刻に TAL 配下で VRP Entries に影響するオブジェクトが 1 件のみ追加・失効したこと，などが主な候補である．両者とも NTP 同期・同一バージョン・同一手順で同時実行されており，差分が出た点については今後の研究にて詳細に調査する．今回の検証結果において重要な要素として，以降の時刻における曲線の折れ曲がり位置や漸減特性，0 件に到達する閾値が，OS を越えて高い再現性で一致している点を分析対象とした．

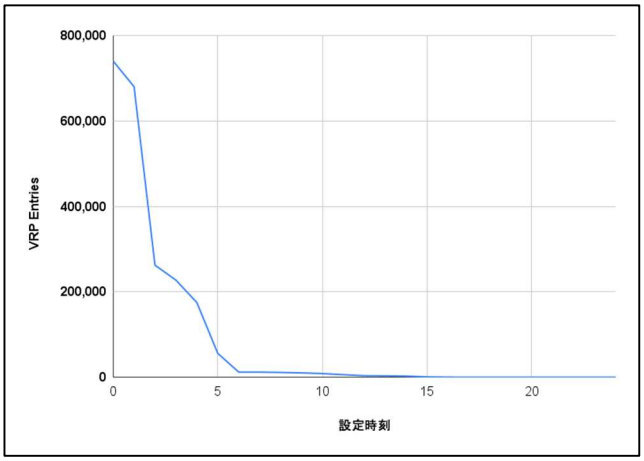

図 2　設定時刻と VRP Entries の相関グラフ（Ubuntu，1 時間毎）

Figure 2　Correlation graph between configured time and VRP Entries (Ubuntu, hourly).

表 1　各設定時刻と VRP Entries（Ubuntu，1 時間毎）

Table 1　VRP Entries at each configured time (Ubuntu, hourly).

設定時刻	VRP Entries
0 時間前	740,846
1 時間前	680,751
2 時間前	262,598
3 時間前	227,627
4 時間前	175,043
5 時間前	56,001
6 時間前	12,106
7 時間前	12,097
8 時間前	11,351
9 時間前	10,256
10 時間前	8,639
11 時間前	6,024
12 時間前	3,687
13 時間前	3,319
14 時間前	2,681
15 時間前	902
16 時間前	202
17 時間前	0
18 時間前	0

19 時間前	0
20 時間前	0
21 時間前	0
22 時間前	0
23 時間前	0
24 時間前	0

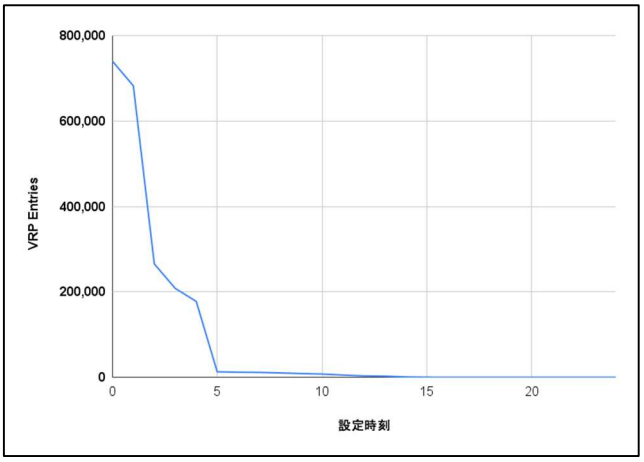

図 3　設定時刻と VRP Entries の相関グラフ（OpenBSD, 1 時間毎）

Figure 3　Correlation graph between configured time and VRP Entries (OpenBSD, hourly).

表 2　各設定時刻と VRP Entries（OpenBSD，1 時間毎）

Table 2　VRP Entries at each configured time (OpenBSD, hourly).

設定時刻	VRP Entries
0 時間前	740,845
1 時間前	682,954
2 時間前	265,785
3 時間前	208,332
4 時間前	177,954
5 時間前	13,083
6 時間前	12,122
7 時間前	11,668
8 時間前	10,388
9 時間前	8,834
10 時間前	7,763
11 時間前	5,310
12 時間前	3,381
13 時間前	2,743
14 時間前	925
15 時間前	213
16 時間前	0
17 時間前	0
18 時間前	0
19 時間前	0
20 時間前	0
21 時間前	0
22 時間前	0
23 時間前	0
24 時間前	0

4. まとめ

本研究では，rpki-client を用いて，OS の違いと時刻同期の有無が Validation 結果に与える影響を実証的に比較した．対象 OS は OpenBSD 7.5 と Ubuntu 22.04 LTS，ツールは同一バージョンの rpki-client 8.6 で統一し，標準出力に現れる統計値と実行ログを収集・比較した．両環境とも NTP により現在時刻を正確に同期し，NTP サーバは NICT の”ntp.nict.jp”へ統一した．

検証 1（OS 間比較）では，同一時刻に同時実行した条件下で，最終的な VRP Entries は一致し，OS の違いが VRP Entries そのものに恒常的な差を生じさせないことを確認した．一方で，処理時間の内訳（system / user）や validated cache の反映・クリーンアップ統計，ログに現れるエラーには差異が観測され，取得実装（rsync / OpenRsync）やファイルシステム挙動等の寄与が示唆された．これらは VRP Entries の一致性に決定的な影響は与えなかったが，運用特性の違いとして今後の詳細な分析が必要である．

検証 2（時刻影響評価）では，基準時刻（0 時間前）から 24 時間前までを 1 時間刻みで実行し，各回の VRP Entries を取得した．その結果，時刻を過去へ戻すほど VRP Entries は減少し，2 時間前付近で約 3 分の 1 へ急減する転移点，11 時間前付近での急激な低下，16〜17 時間前で 0 件に至るという挙動が，OS を越えて高い再現性で現れた．0 時間前の結果は Ubuntu が 740,846 件，OpenBSD が 740,845 件と 1 件の差が生じた．0 時間前の 1 件差については，(i) 実行開始の秒単位の差に伴うリポジトリ側の境界更新との競合，(ii) 取得経路・実装差（Ubuntu では rsync，OpenBSD では OpenRsync）に伴う微小な反映順序の違い，(iii) 同一時刻に TAL 配下で VRP Entries に影響するオブジェクトが 1 件のみ追加・失効したこと，などが主な要因の候補である．両 OS とも NTP 同期・同一バージョン・同一手順で同時実行されており，差分が出た点については今後の研究にて詳細に調査する．今回の検証結果において重要な要素として，以降の時刻における曲線の折れ曲がり位置や漸減特性，0 件に到達する挙動が，OS を越えて高い再現性で一致している点を分析対象とした．

また，運用上の推奨としては，(a) NTP 冗長化と同期失敗の即時検知，(b) 時刻異常の監視，(c) RRDP / rsync 到達性とフェイルオーバの常時監視，(d) validated cache の運用ポリシー（取得タイミングの固定・スナップショット保持・

独立キャッシュ運用）を挙げる．検証 2 の結果より，特に OpenBSD における運用では境界となる時刻付近で安全側に倒れる傾向があり，時刻健全性の逸脱を早期に検知・是正しやすい特徴が観測できた．

本研究における課題点として，取得方式の主対象が rsync であること，rsync と OpenRsync の実装差やファイルシステム要因の寄与度を厳密に切り分けていないこと，時刻ずれ評価におけるネットワーク経路の影響（RRDP フェイルオーバ等）を定量的に評価していないことが挙げられる．これらを踏まえ，今後は（i）取得実装差が処理時間・回復動作・キャッシュ反映に与える影響の精査，（ii）rpki-client に実装されている，-j オプションの出力に基づく可視化・差分抽出ツールの整備，（iii）Routinator 等の別実装による再現検証を通じて，RPKI 検証基盤の信頼性向上と具体的なベストプラクティスの確立を進める．

参考文献

[1] Y. Rekhter, Ed., T. Li, Ed., S. Hares, Ed. “RFC 4271 - A Border Gateway Protocol 4 (BGP-4)” 2006.

[2] K. Sriram, D. Montgomery, US NIST, D. McPherson, E. Osterweil, Verisign, Inc., B. Dickson "RFC 7908 - Problem Definition and Classification of BGP Route Leaks" 2016.

[3] C. Alaettinoglu, USC/Information Sciences Institute, C. Villamizar, Avici Systems, E. Gerich, At Home Network, D. Kessens, Qwest Communications, D. Meyer, University of Oregon, T. Bates, Cisco Systems, D. Karrenberg, RIPE NCC, M. Terpstra, Bay Networks “RFC 2622 – Routing Policy Specification Language(RPSL)” 1999.

[4] L. Daigle, VeriSign, Inc. “RFC 3912 - WHOIS Protocol Specification” 2004.

[5] Alain Durand “Resource Public Key Infrastructure (RPKI) Technical Analysis (OCTO-014)”, 2020.

[6] RIPE NCC, “RIPE NCC IRR Database Non-Authoritative Route Object Clean-up (RIPE-731)”, 2019.

[7] B. Gorman, American Registry for Internet Numbers (ARIN), “Take Action Now to Prepare for the ARIN-NONAUTH IRR Retirement,” 2022.

[8] M. Lepinski, S. Kent, BBN Technologies “RFC 6480 - An Infrastructure to Support Secure Internet Routing” 2012.

[9] Cisco Systems, J. Scudder, Juniper Networks, D. Ward, R. Bush, Internet Initiative Japan, R. Austein, Dragon Research Labs "RFC 6811 - BGP Prefix Origin Validation" 2013.

[10] 一般社団法人日本ネットワークインフォメーションセンター(JPNIC), “RPKI の ROA を使ったインターネットにおける不正経路への対策ガイドライン" 2024.

[11] T. Bruijnzeels, O. Muravskiy, RIPE NCC, B. Weber, Cobenian, R. Austein, Dragon Research Labs “RFC 8182 - The RPKI Repository Delta Protocol (RRDP)” 2017.

[12] G. Huston, APNIC, S. Weiler, W3C/MIT, G. Michaelson, S. Kent, Unaffiliated, T. Bruijnzeels, NLnet Labs “RFC 8630 - Resource Public Key Infrastructure (RPKI) Trust Anchor Locator” 2019.

[13] R. Austein, Arrcus, Inc., G. Huston, APNIC, S. Kent, Independent, M. Lepinski, New College Florida “RFC 9286 - Manifests for the Resource Public Key Infrastructure (RPKI)” 2022.

[14] G. Huston, G. Michaelson, R. Loomans, APNIC “RFC 6487 - A Profile for X.509 PKIX Resource Certificates” 2012.

[15] R. Bush, Internet Initiative Japan, R. Austein, Dragon Research Labs “RFC 8210 - The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 1” 2017.

[16] Asia Pacific Network Information Centre “Use of Route Object Validation for Japan (JP)” 2025.

[17] D. Mills, U. Delaware, J. Martin, Ed., ISC, J. Burbank, W. Kasch, JHU/APL “RFC 5905 - Network Time Protocol Version 4: Protocol and Algorithms Specification” 2010.

[18] 一般社団法人日本ネットワークインフォメーションセンター(JPNIC), “JPNIC News & Views vol.1469(2017 年 1 月 16 日発行)" 2017.