IXのルートサーバーにおける RPKIの取り組み

JANOG47

インターネットマルチフィード 後藤 / 萩原

この発表について

- 背景
  - 2020年は世界中でRPKIの導入が進み、IXのルートサーバーでも対応が求められる
  - JPNAPは2020年11月、商用のルートサーバーにRPKI ROVを導入した
  - ICPやCSPなど他事業者と比較するとRPKIの導入障壁は低いが、やはり検討は必要

- モチベーション
  - 手探りながら本番環境にROVを適用した足跡を共有し、RPKI推進に向けた議論を行いたい
  - (ルートサーバーの仕組みを紹介したい)

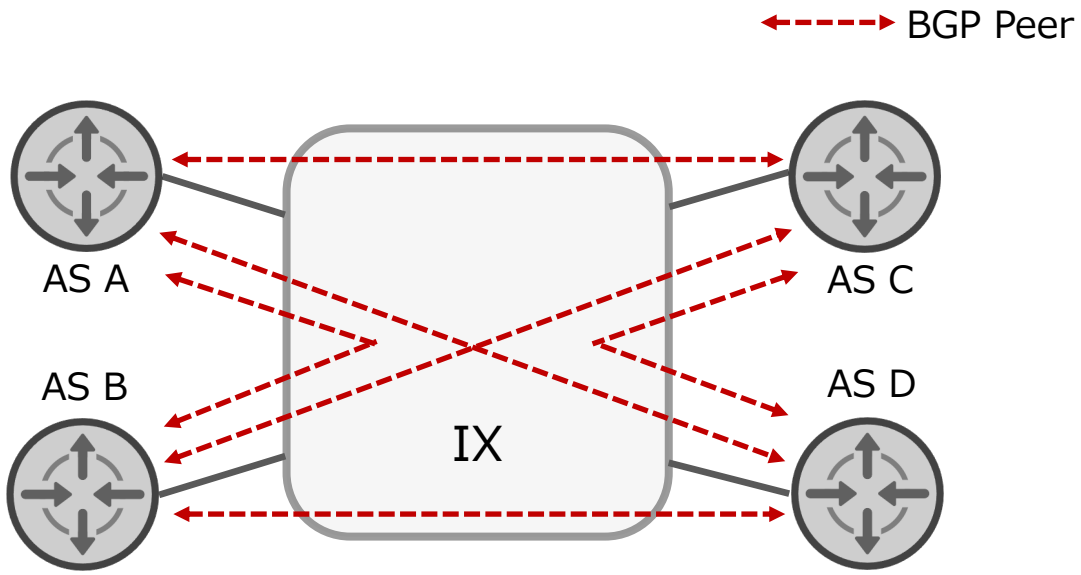
- 発表者
  - インターネットマルチフィード株式会社 JPNAPチーム ルートサーバー開発担当 2名
  - 後藤 成聡 Nasato GOTO goto@mfeed.ad.jp
  - 萩原 昂 Noboru HAGIWARA hagiwara@mfeed.ad.jp

ルートサーバー

ルートサーバー (Route Server, RS)

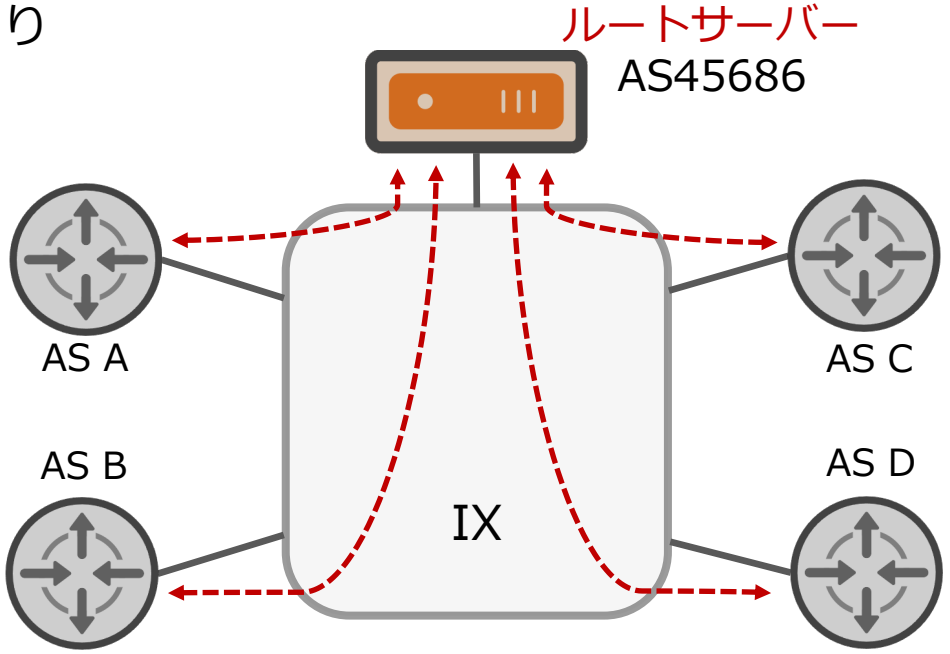
- IX上でBGPによる経路交換を効率化する仕組み
- RSとピアすることで、他の全てのRSピアと自動的に経路を交換できる (マルチラテラル・ピアリング)

RSなし



- トラフィックを交換したいASと直接ピアして経路を交換 (バイラテラル・ピアリング)
- 経路制御の自由度は高い

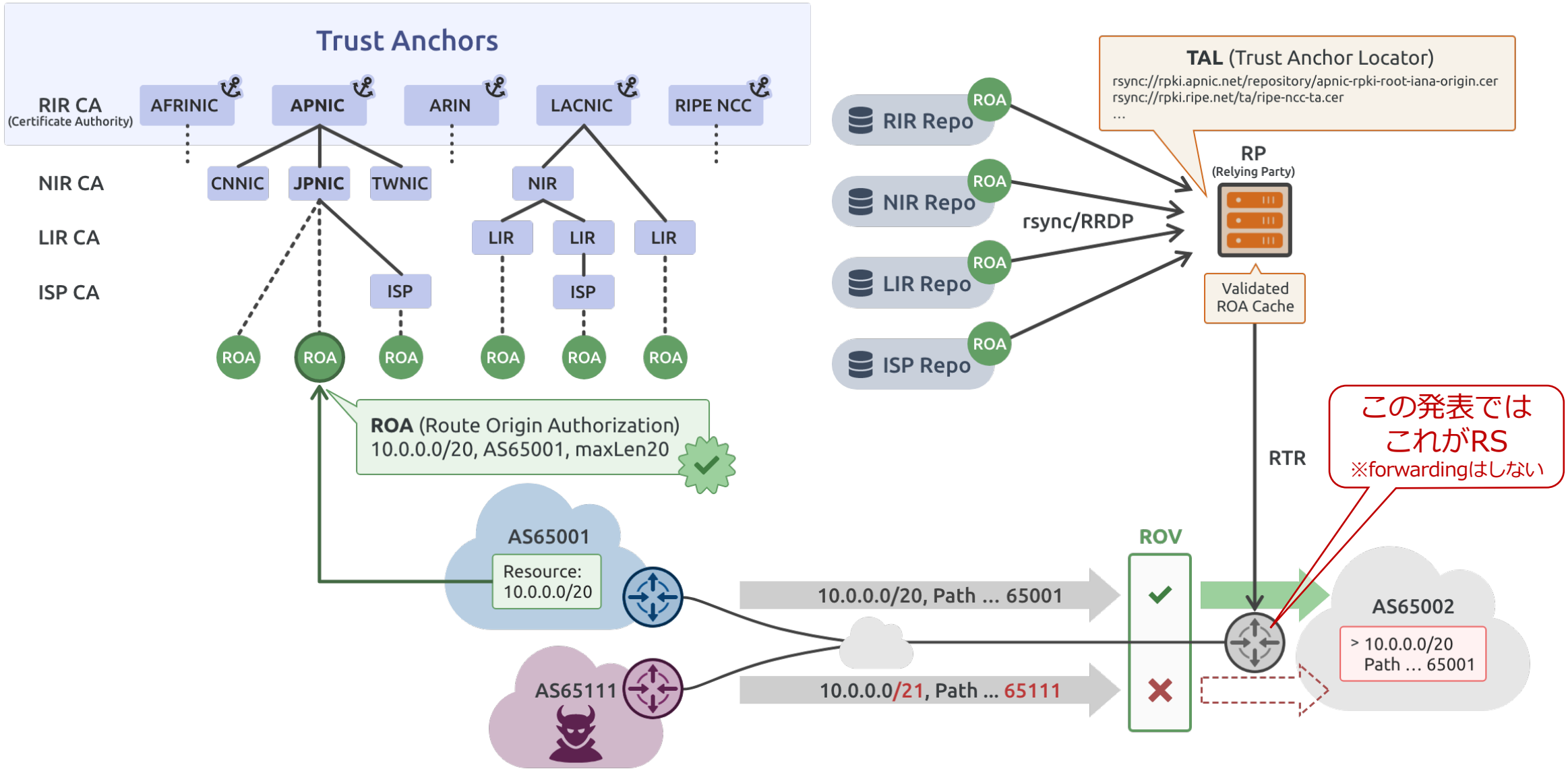
RSあり



- 実トラフィックはRSを経由しない (= Control Planeのみ提供)
- Path Attribute (AS_PATH, MED等) を透過的に扱う
- 経路制御はRSが提供する機能に依存 (BGP Communityによる制御)

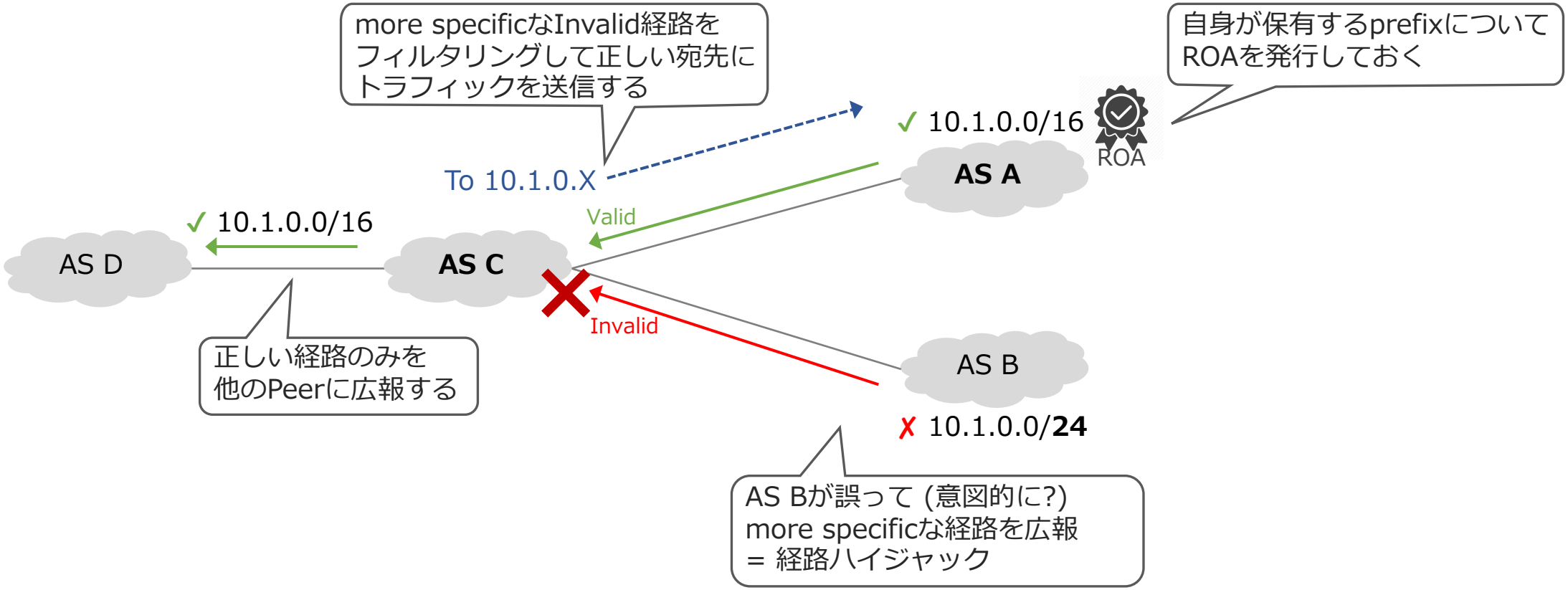
RPKI ROV 概要

本発表ではRPKI自体の説明は省きます。
よろしければ プログラムページ の事前資料をご覧ください。



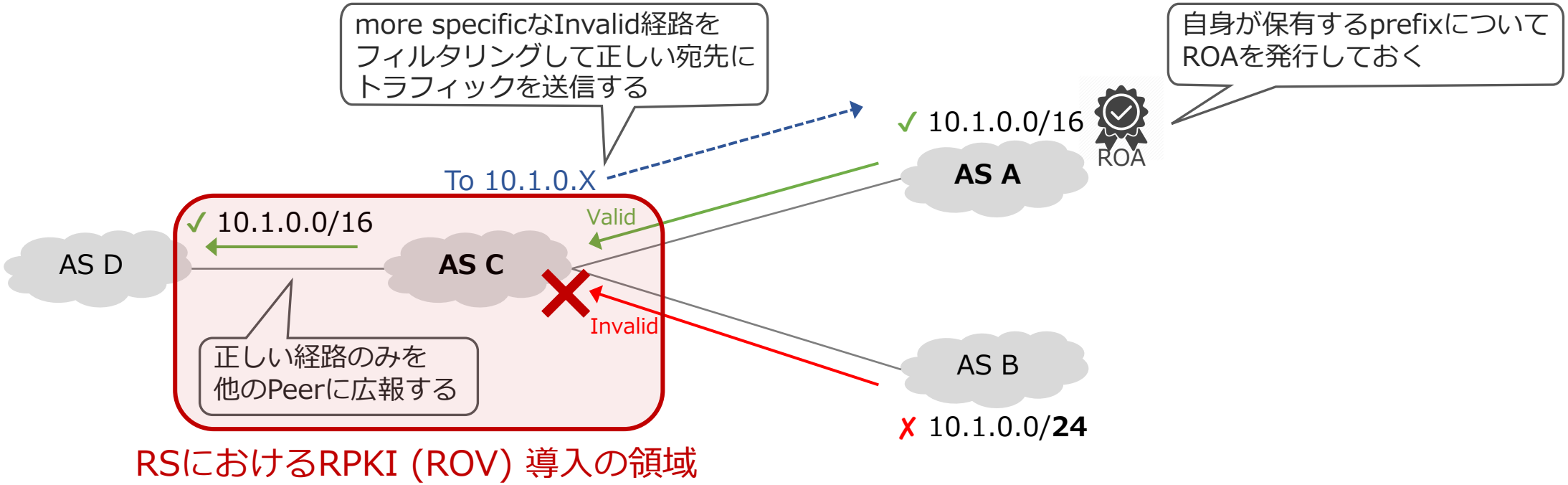
RPKIの意義

- (1) ROAを発行することで自身の経路がハイジャックされた際の影響を軽減する ※(2)がある前提
  - 「AS AはROAによりAS Bが不正に経路広報した状況下でAS Cからのトラフィックを正常に受信できる」
- (2) ROVを導入することでハイジャックされた経路の拡散や当該経路へのトラフィックの送信を防止する
  - 「AS Cは不正な経路をAS Dに広報せず、正しい広報元であるAS Aにトラフィックを送ることができる」



RSにRPKIを導入する

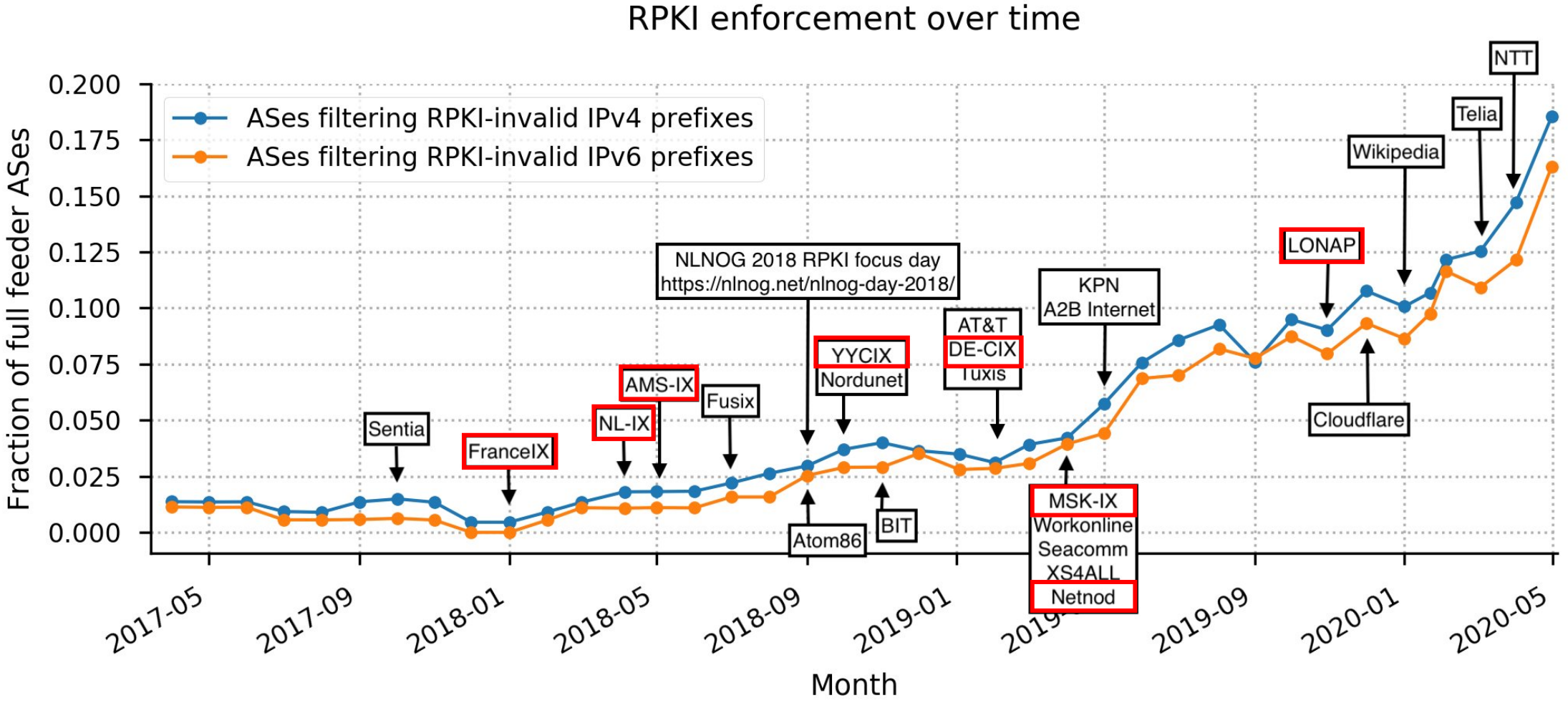
- (1) ROAを発行することで自身の経路がハイジャックされるリスクを低減する ※(2)がある前提
  - 「AS AはROAによりAS Bが不正に経路広報した状況下でAS Cからのトラフィックを正常に受信できる」
- (2) ROVを導入することでハイジャックされた経路の拡散や当該経路へのトラフィックの送信を防止する
  - 「AS Cは不正な経路をAS Dに広報せず、正しい広報元であるAS Aにトラフィックを送ることができる」



RPKI ROV for RS
- クライアントから受信した経路に対しROVを行い、その結果に基づいて経路のフィルタリングを行う
- 経路を広く伝搬する性質を持つためRPKI ROVの導入は重要

世界のIXのROV導入状況

- 2018年より徐々に、主に欧州のIXを中心にROVの導入が進められてきた (図の赤い囲み)



https://twitter.com/JobSnijders/status/1256326712347881473

世界の(IXの)ROV導入状況

- APIX (Asia Pacific地域のIX) メンバーも徐々に導入を進める
  - 32 IXのうち4 IXが実施 (Web公開情報や伝聞による、2021/01調査)
    - BKNIX : 2019/03~
    - HKIX : 2020/08~
    - IX Australia : 2020/09~
    - TWIX : 2020/??~
- IX以外にもISPやCSPでROA作成/ROV導入の報告が多数
  - ROUTING SECURITY: RPKI UPDATE Q2/20
    - Teliaのレポート、2020年多くのTier1がInvalidをdrop
  - 安全なインターネット ルーティングの実現に向けた Google の取り組みの拡大
    - 保有するIP空間の99%をROA登録し、2021年にROVを導入予定
  - How AWS is helping to secure internet routing
    - 保有するIP空間の99%をROA登録し、全AS16509 POPでInvalidをdrop

Is BGP safe yet? No.

NAME	TYPE	DETAILS	STATUS
Telia	transit	signed + filtering	safe
Cogent	transit	signed + filtering	safe
GTT	transit	signed + filtering	safe
NTT	transit	signed + filtering	safe
Hurricane Electric	transit	signed + filtering	safe
TATA	transit	signed + filtering	safe
PCCW	transit	signed + filtering	safe
RETN	transit	partially signed + filtering	safe
Cloudflare	cloud	signed + filtering	safe
Amazon	cloud	signed + filtering	safe
Netflix	cloud	signed + filtering	safe
Wikimedia Foundation	cloud	signed + filtering	safe
Scaleway	cloud	signed + filtering	safe
Telstra International	transit	signed	partially safe
AT&T	ISP	signed + filtering peers only	partially safe
Google	cloud	signed	partially safe

https://isbgpsafeyet.com/

インターネット全体でRPKIの機運が高まっている
(今がチャンス!!)

導入に向けたステップ (= 発表目次)

1.設計
ポリシー決め、技術選定、パラメーター検討

2.検証
設計した技術の要素検証、特にROVやRelying Partyとの接続など

3.導入
本番環境への適用に至るまでのタイムライン、顧客周知や導入結果の確認など

4.運用
監視やオペレーション、ユーザー向けツール

導入に向けたステップ

1.設計
ポリシー決め、技術選定、パラメーター検討

2.検証
設計した技術の要素検証、特にROVやRelying Partyとの接続など

3.導入
本番環境への適用に至るまでのタイムライン、顧客周知や導入結果の確認など

4.運用
監視やオペレーション、ユーザー向けツール

設計: ポリシーの決定

- ROVの結果 (Valid/NotFound/Invalid) に応じた経路フィルタリングをどうするか？
  - ≒ Invalidはrejectしてよいか？

- コミュニティやユーザーと議論
  - Strategy for deploying RPKI ROV to Route Server on IX, 2019/09 APNIC48
    - JPNAP「Invalidの取り扱いを経路の送受信者が選べるようにするのはどうか？」
    - フィードバック「顧客にとって複雑なポリシーは避けたほうがよい、Invalidはdropでよい」
  - 弊社ユーザー会併設イベントでアンケート, 2019/10
    - 半数以上のユーザーが「Invalidは広報しなくてよい」と回答

- 2020年多くの事業者がROVを有効化していたこともあり、既に「Invalidな経路にタグ付けをして広報する」フェーズは終了し、「Invalidな経路は落とす」運用が一般的になっていると認識

Valid, NotFound → accept, Invalid → reject
に決定

設計: BGPd選定

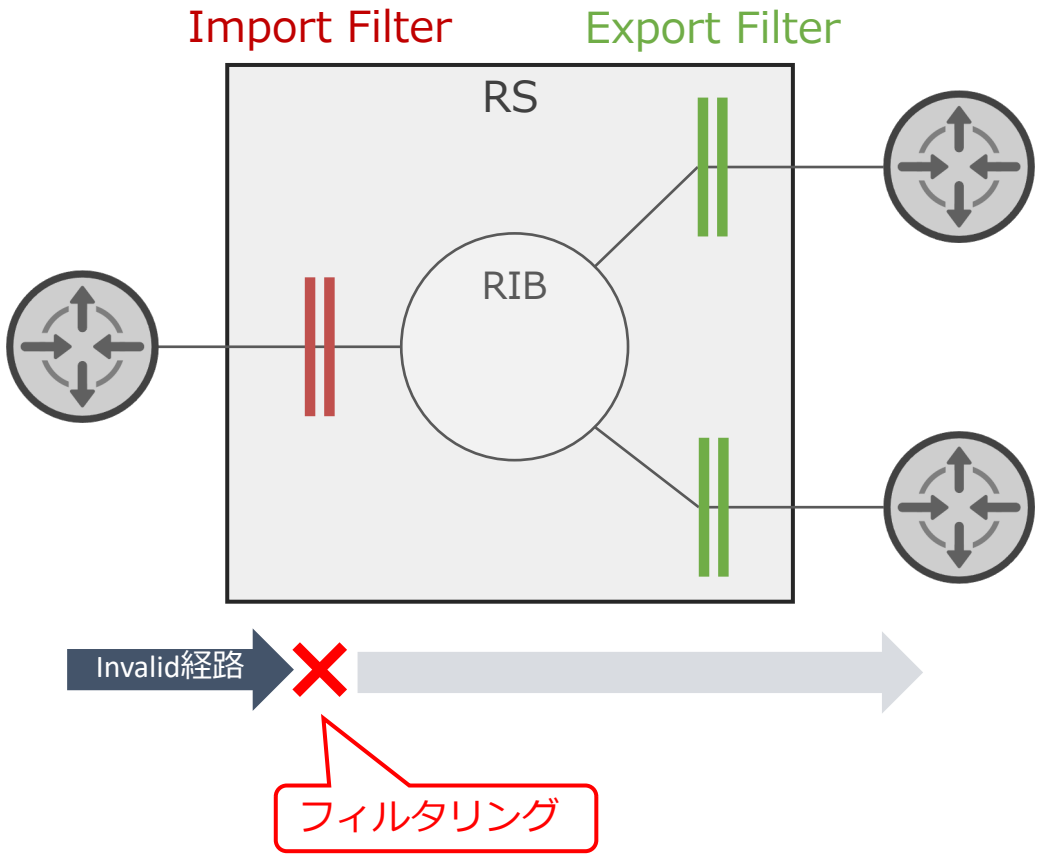
- 前提: それまで利用していた一つのBGPdは古すぎてRPKIに対応していなかった
  - (RPKIを使うためにBGPdの更改という裏テーマがあった)
- OSS BGPd RPKI機能実装状況

	BIRD1.0	BIRD2.0	GoBGP	OpenBGPD	FRR
RTR対応	No ※configにstaticでVRPを定義	Yes	Yes	No ※rpki-clientを併用	Yes
ROVフィルタリング	Yes	Yes	Yes	Yes	Yes

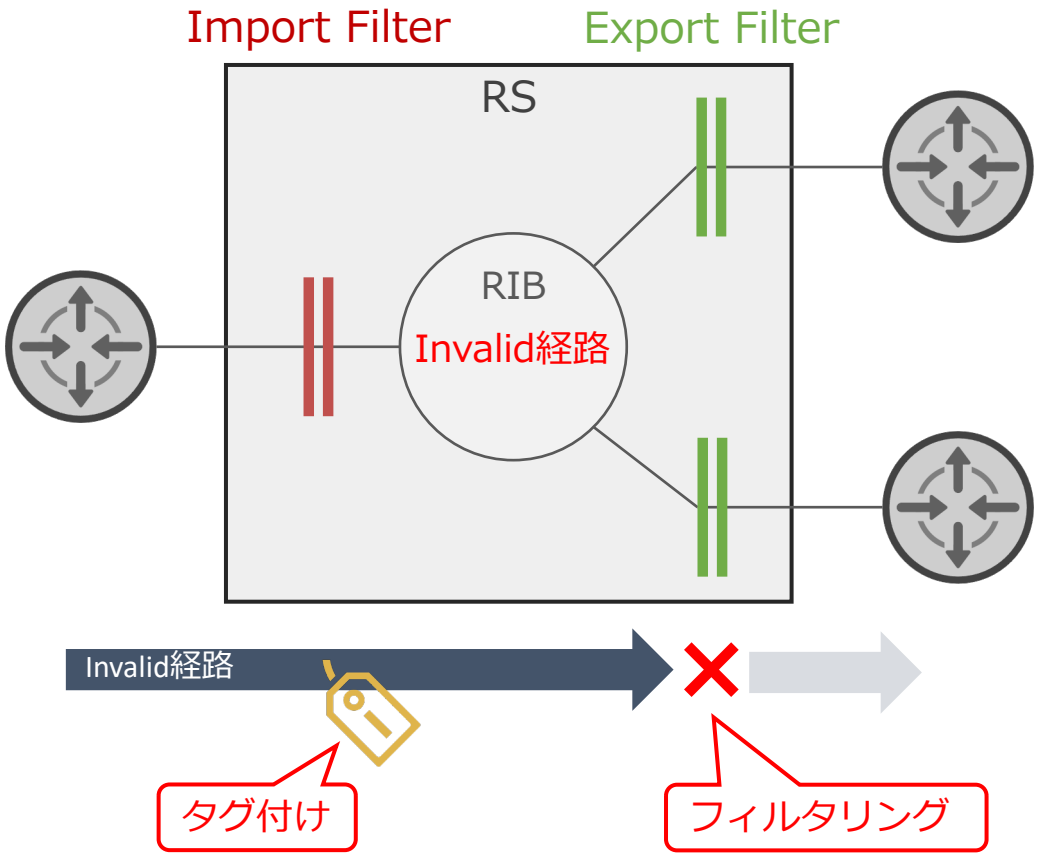
- BIRD2.0を選択
- 選択の背景
  - RPKI機能はいずれも最低限要件を満たしているが、その上で利用者の多さは重要
    - そもそも多くのIX (体感で70%以上) がルートサーバー用途でBIRDを選択する
    - JPNAPでも2013年頃からBIRD1.0を利用していた実績あり
  - 周辺ツールが豊富
    - APIサーバー、exporter、Looking Glassなど
  - BIRD1.0と比較すると利用IXはまだ少ないが、開発者はBIRD2.0への移行を推奨

設計: ROVフィルタリング実装戦略

方式1: 受信したInvalid経路をすぐにフィルタリングする



方式2: 受信したInvalid経路にCommunityを付与し、
広報時にそれに従いフィルタリングする



※BIRD2.0ではいずれも実現可能

設計: ROVフィルタリング実装戦略

方式1: 受信したInvalid経路をすぐにフィルタリングする

- ✓ パフォーマンスが (多少) 良い
  ∵ Invalid経路がインストールされないため
- ✓ 「条件を満たさない経路はreject」という統一されたポリシーを適用することができる
  → configの設計をシンプルに保つ
- ✗ 前述の「Invalidな経路にタグ付けをして広報する」はできない

方式2: 受信したInvalid経路にCommunityを付与し、広報時にそれに従いフィルタリングする

- ✓ Invalid経路の広報が可能
  → 実験用途など特殊な要望に応じることができる
- ✗ RIBにInvalid経路がインストールされる
  → 多くの場合においてInvalid経路の広報は不要
- ✗ RPKI以外の経路フィルターとの整合性の確保が必要
  → IRRフィルターに弾かれた経路はどうする？
  → configの見通しが悪くなる

ROV導入済みの10 IXの実装方法を調査

- 方式1: **7 IX** (DE-CIX, LINX, LONAP等)
- 方式2: **3 IX** (AMS-IX, BKNIX, France-IX)

決定したポリシーをシンプルに実現可能な方式1を採用

※ただし今後ユーザーに利するところがあると判断した場合には方式2への転換も検討する

設計: Relying Party (Validator) 選定

- Routinator3000 を利用
  - 2018~, Github Star 200+
- 選定理由
  - シングルバイナリでインストールが容易
  - RTRサーバー内蔵
  - Web APIの提供、exporter内蔵でGrafana連携容易
  - 開発が活発
  - APRICOT2020のRPKI Deployathonでも良い評価

- 構成
  - Relying Party実装の冗長化はなし、現時点ではRoutinator3000のみ利用
    - インフラコード化 (ansible化) 稼働や運用コスト削減のため
    - (将来的にやりたいが最優先事項ではない...?)
  - Relying PartyはRSと近いネットワークセグメント (監視網) に配置
    - JPNAP{東京|大阪|福岡} それぞれ2インスタンスずつ、RSは2インスタンスとRTR接続
- 議論
  - IXセグメント上でRTR接続を提供するサービスって使いたいと思いますか？
    - 安心度としては「自社網内 > IX > public」くらいのイメージ、+ssh等secure化でさらに◎

(参考) Relying Party (Validator) 比較

		RIPE NCC RPKI Validator	Routinator3000	FORT	OctoRPKI	rpki-client(-portable)
概要	メンテナー	RIPE NCC	NLnet Labs	FORT Project (LACNIC + NIC.MX)	Cloudflare	個人 (OpenBSD Projectの一部)
	開始	2011	2018	2018	2019	2020
	Github Star数	50+	200+	20+	100	10 ※portable版
	言語	Java	Rust	C	Go	C
機能	RTRサーバー同梱	No (同repo内で分割)	Yes	Yes	No (GoRTR対応)	No (GoRTR対応)
	Secure RTR	No	Yes ※ssh proxyが必要	Yes	- (GoRTR対応)	- (GoRTR対応)
	SLURM対応	Yes	Yes	Yes	- (GoRTR対応)	- (GoRTR対応)
付加価値	WebAPI提供	Yes	Yes	No	No	No
	Prometheus exporter/metrics	Yes	Yes	No	Yes	No
	ドキュメント充実 ※個人の主観	☺	☺	😐	😐	☹
備考	-	2021/07 EOL				

※2021/01時点

(参考) Relying Party毎にVRP数が異なる？

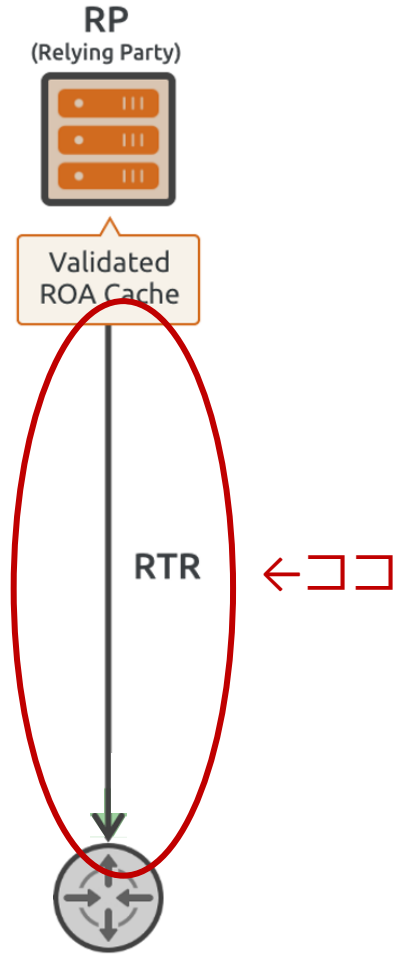
- APNIC Blogの記事によるとRelying Partyによって出力されるVRP (Validated ROA Payloads) 数が異なる
  - RoutinatorとRIPE NCC Validatorは同数で最多のVRP数、当時v4が114,961、v6が19,307
  - FORTはこの2つに対し1,2個少ない (取得タイミングによる誤差?)
  - OctoRPKIは2つに対し~1,200程度少ない
- 同じ検証を日を変えて何度か実施した

測定日		RIPE NCC RPKI Validator	Routinator3000	FORT	OctoRPKI	rpki-client(-portable)
2021/01/14	v4	181,788	181,788	181,788	180,516	181,788
	v6	30,601	30,600	30,600	29,901	30,600
2021/01/17	v4	182,010	182,010	182,010	180,755	181,939
	v6	30,646	30,646	30,645	29,957	30,634
2021/01/20	v4	182,414	182,414	182,414	181,170	182,415
	v6	30,941	30,941	30,941	30,248	30,942

- 検証結果
  - APNIC Blog記事と同様、OctoRPKIのみVRP数が2,000弱少ない
  - 他のRelying Partyは多少の誤差はあるものの、ほぼ同程度のVRP数
    - タイミングイシューの範疇か
  - 原因の特定には至っていない (ご存知の方がいたら教えてください)

設計: パラメーター

- RTR関連
  - 更新間隔: 1時間
    - BGPdからRelying Party (キャッシュサーバー)に対し VRPの更新要求 (Serial Query) を行う間隔
    - RFC8210 によると推奨値 1時間
  - 失効時間: 36時間
    - 更新に失敗した際にダウンロードしたVRPを保持しておく時間
    - 最大1.5日、更新に失敗することを許容する設計
      - その猶予の中でトラシューして何とかする、という考え
    - RFC8210 によると推奨値2時間
      - 実際に運用をする中でRTR接続の問題が起きたことは現状ほとんどなく、推奨値に近づけるように見直す予定

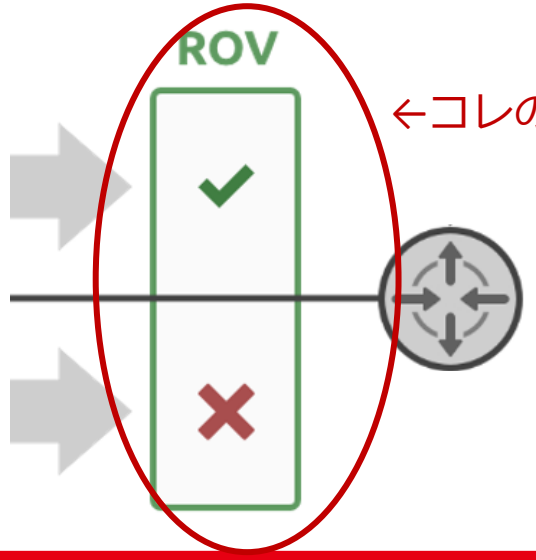


設計: パラメーター

前提: IRRベースの経路フィルター更新は現状1回/1日

※これ自体をより高頻度化することは別途検討中

- ROV re-validation
  - 既にAdj-RIBs-INに受信済みの経路に対し、更新されたVRPで再度ROVを行う
    - RFC6811 *“When a mapping is added or deleted, the implementation MUST re-validate any affected prefixes and run the BGP decision process if needed. ”*
  - BIRD2.0はv2.0.7時点で自動re-validationに非対応
  - 経路フィルター更新のタイミングで明示的にre-validationするコマンドを実行
    - `‘birdc reload in all’`
    - 作用としてはROUTE-REFRESHでクライアントから経路を再受信しROVを再適用する



←コレの再実施タイミング

導入に向けたステップ

1.設計
ポリシー決め、技術選定、パラメーター検討

2.検証
設計した技術の要素検証、特にROVやRelying Partyとの接続など

3.導入
本番環境への適用に至るまでのタイムライン、顧客周知や導入結果の確認など

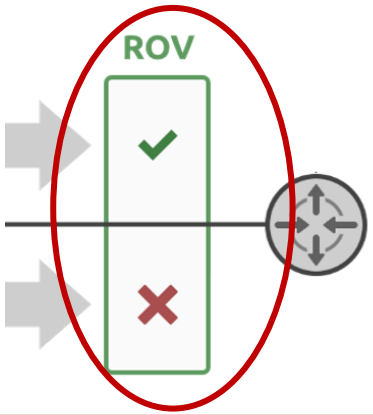
4.運用
監視やオペレーション、ユーザー向けツール

検証: BGPd/ROV

- 観点: BGPdがRFC6811のアルゴリズムに従いROV判定を実施可能であること確認する
- 検証項目
  - Invalid origin判定
  - Invalid length判定
  - AS0判定
- (この表1枚作ってあると何かと便利)

No	ROA Prefix/MaxLen	ROA OriginAS	BGP Route Prefix	BGP Route OriginAS	ROV Result
1	10.101.0.0/24-24	65001	10.101.0.0/24	65001	Valid
2	-	-	10.102.0.0/24	65001	NotFound
3	10.103.0.0/24-24	65111	10.103.0.0/24	65001	Invalid
4	10.104.0.0/24-24	65001	10.104.0.0/23	65001	NotFound
5	10.105.0.0/24-24	65001	10.105.0.0/24	65001	Valid
6	10.106.0.0/24-24	65001	10.106.0.0/25	65001	Invalid
7	10.107.0.0/23-25	65001	10.107.0.0/22	65001	NotFound
8	10.108.0.0/23-25	65001	10.108.0.0/23	65001	Valid
9	10.109.0.0/23-25	65001	10.109.0.0/24	65001	Valid
10	10.110.0.0/23-25	65001	10.110.0.0/25	65001	Valid
11	10.111.0.0/23-25	65001	10.111.0.0/26	65001	Invalid
12	10.112.0.0/23-25	65111	10.112.0.0/22	65001	NotFound
13	10.113.0.0/23-25	65111	10.113.0.0/24	65001	Invalid
14	10.114.0.0/23-25	65111	10.114.0.0/26	65001	Invalid
15	10.115.0.0/23-25	0	10.115.0.0/22	65001	NotFound
16	10.116.0.0/23-25	0	10.116.0.0/24	65001	Invalid
17	10.117.0.0/23-25	0	10.117.0.0/26	65001	Invalid

ココの正しさ
を検証↓

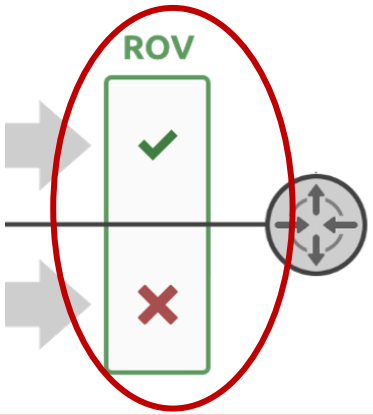


検証: BGPd/ROV

- 観点: BGPdがRFC6811のアルゴリズムに従いROV判定を実施可能であること確認する
- 検証項目
  - 同一prefix/複数ROAの判定

No	ROA Prefix/MaxLen	ROA OriginAS	BGP Route Prefix	BGP Route OriginAS	ROV Result
1	10.103.0.0/24-24	65001	10.103.0.0/24	65001	Valid
	10.103.0.0/24-24	65001			
2	10.104.0.0/24-24	65001	10.104.0.0/24	65001	Valid
	10.104.0.0/24-24	65111			
3	10.105.0.0/23-25	65001	10.105.0.0/24	65001	Valid
	10.105.0.0/24-24	65111			
4	10.106.0.0/23-23	65001	10.106.0.0/24	65001	Invalid
	10.106.0.0/23-25	65111			

ココの正しさ
を検証↓



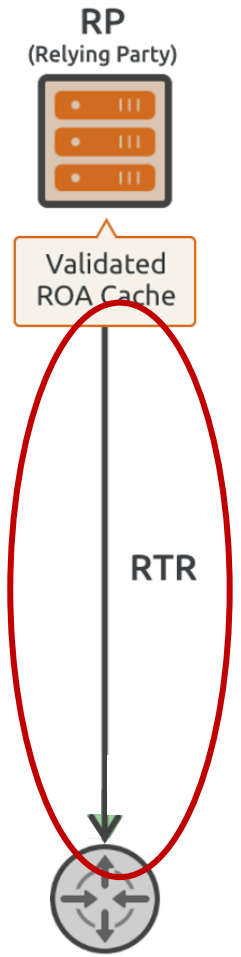
（参考）BGPd検証の自動化

- BGPd検証の約50%は自動化、ラップトップ上で試験が可能
  - GoBGPのScenario Test を拡張 (今でも有り難く有効活用させてもらっています)
  - RSはData Planeを提供しないため簡単にコンテナ化が可能、通常のルーターと比較して検証を自動化しやすい
- 前述のROV検証の試験シナリオイメージ
  - 各経路に付与されているROV結果を示すLarge Communityをチェックしている

```
# 検証対象のルートサーバー（BIRD2.0）とクライアント（GoBGP）コンテナを起動
rs = BirdContainer(name='rs', asn=45686, ip_addr='210.173.176.213')
g1 = GoBGPContainer(name='g1', asn=65001, router_id='210.173.176.1', ip_addr='210.173.176.1')

# クライアントから経路を広報
g1.add_route('10.101.0.0/24', aspath=[65001])
g1.add_route('10.102.0.0/24', aspath=[65001])
g1.add_route('10.103.0.0/24', aspath=[65001])

# 以降、ROV判定結果をテスト

# No: 1
# Expected RPKI ROV Result: Valid
lc = [p['bgp']['large_communities'] for p in rs.get_adj_rib_in(g1) if p['prefix'] == '10.101.0.0/24']
assertTrue([45686, 1000, 1] in lc[0])

# No: 2
# Expected RPKI ROV Result: NotFound
lc = [p['bgp']['large_communities'] for p in rs.get_adj_rib_in(g1) if p['prefix'] == '10.102.0.0/24']
assertTrue([45686, 1000, 2] in lc[0])

# No: 3
# Expected RPKI ROV Result: Invalid
lc = [p['bgp']['large_communities'] for p in rs.get_adj_rib_in(g1) if p['prefix'] == '10.103.0.0/24']
assertTrue([45686, 1000, 4] in lc[0])
assertTrue([45686, 1101, 13] in lc[0])
```

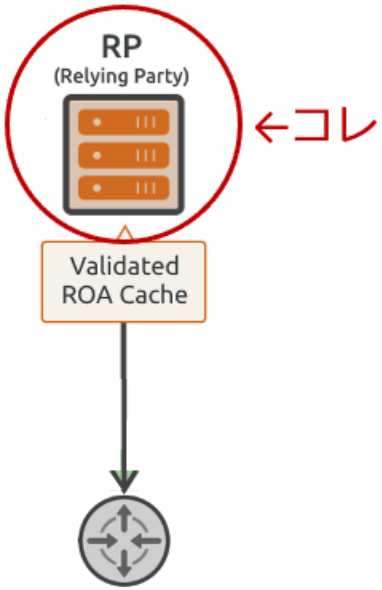
検証: BGPd/RTR接続

- 観点: BGPdとRelying Party (キャッシュサーバー) とのRTR接続動作を把握する
  - プロトコルベースというよりは実際の動作ベースで検証を行った
- 検証項目
  - 接続/切断
    - 複数のキャッシュサーバーへRTR接続ができるか
    - コマンドによるRTRセッションのリセットや再接続ができること
    - RTRセッションの切断や接続ログの確認
  - パラメーター
    - 指定した更新間隔でVRPの再取得が行われること
    - 接続に失敗した際に指定した間隔で接続をリトライすること
  - Expire動作
    - RTRセッション切断後、VRPがExpireするまでの間、既存のROV判定が変わらないこと
    - Expire後にローカルのVRPが消えること
    - Expire後、ROV再判定することでNotFoundになること
  - など



検証: Relying Party

- 観点: Relying Partyが正しくROAの署名検証をしてVRPを生成できるか
- しかし、Relying Party自体の検証はなかなかハードルが高いのではないか...?
  - 自前でCAを建て任意のROAを作成し、Relying Partyを検証する
    - 最近ではOSSのCAとしてKrillがあり以前より取り組みやすさはありそう
    - 2,3年前にDragon Research LabsのCAを試したが、 (私は) そもそもインストールすらできなかった...
- 最低限このくらいは確認
  - ROAのダウンロード、定期更新、ログ出力ができる
  - 他のRelying Party実装と同程度数のVRPを出力できる
  - + VRPをRTRでルーター等にfeedできる (前述のBGPd/RTR接続で代替)
- だからこそ信頼性の高いRelying Partyを選択することが重要か
  - 信頼性が高い ≒ 利用者が多い、開発が活発、Bugfixが早い
  - + Relying Party実装の冗長化をすれば...
- アドバイスあれば是非お願いします!



導入に向けたステップ

1.設計
ポリシー決め、技術選定、パラメーター検討

2.検証
設計した技術の要素検証、特にROVやRelying Partyとの接続など

3.導入
本番環境への適用に至るまでのタイムライン、顧客周知や導入結果の確認など

4.運用
監視やオペレーション、ユーザー向けツール

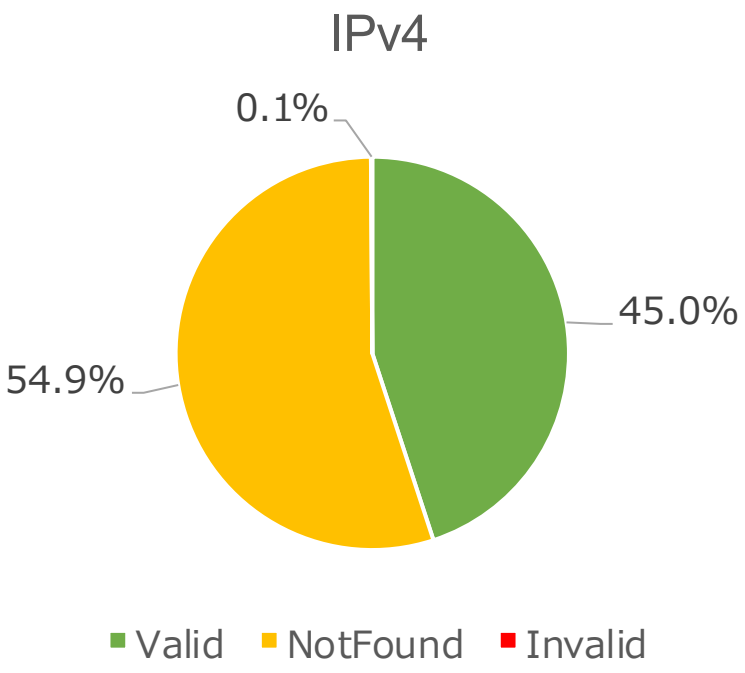
導入: タイムライン

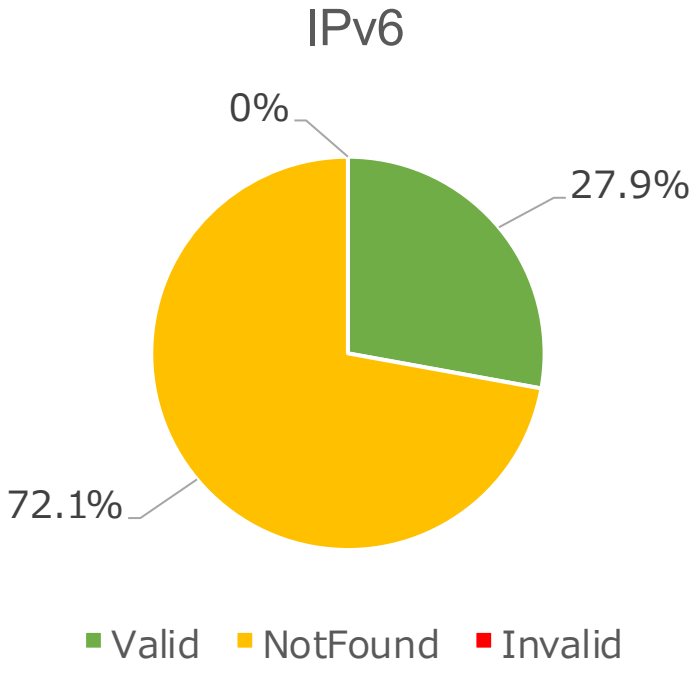
JPNAP東京 道のり

- 2020/01 (10ヶ月前) BIRD2.0/RPKI 設計・検証開始
- 2020/05 (6ヶ月前) ユーザー会でRPKI技術セッションを開催
  - RPKIのおさらいやユーザーAS様の導入事例、世の中のトレンドなどを議論
- 2020/06 (5ヶ月前) JPNAP東京にトライアルでRPKI対応ルートサーバーをデプロイ
  - 3台目のルートサーバーを構築し、ユーザーに協力を募る
    - 周知活動が足りず参加ASをあまり集められませんでした...
    - ご協力いただいたAS様ありがとうございました！
- 2020/10下旬 (3週間前) ユーザー会/メーリングリストでROV導入を周知
  - JPNIC様よりコメント (後述)
  - 反対意見や待ったはなし
    - 改めて国内でもInvalid経路をrejectすることに対する拒否感はない、と理解
- 2020/11上旬 (2週間前) Invalid経路を広報しているユーザーにメール通知
  - 現在acceptされている経路が新たにrejectされるため、ユーザー周知を実施
  - 当時の受信経路にlabでROVを適用、Invalid経路を広報していた13 AS (ほぼ外資) に連絡
  - うち4 ASから返答あり、一部が対処をしてくれた
- 2020/11中旬 BIRD2.0導入、ROV開始

導入: 最近の経路状況

- JPNAP東京 2021/01/17調査
- 経路状況
  - IPv4の経路は45%がValid判定とおよそ半数の経路がROA登録されている
    - それに比べるとIPv6のValid率は低い
    - 例えばJPNIC保有アドレスではIPv6の方がROA登録率は高いけれど…
  - IPv6のInvalid経路が0 (!)





導入に向けたステップ

1.設計
ポリシー決め、技術選定、パラメーター検討

2.検証
設計した技術の要素検証、特にROVやRelying Partyとの接続など

3.導入
本番環境への適用に至るまでのタイムライン、顧客周知や導入結果の確認など

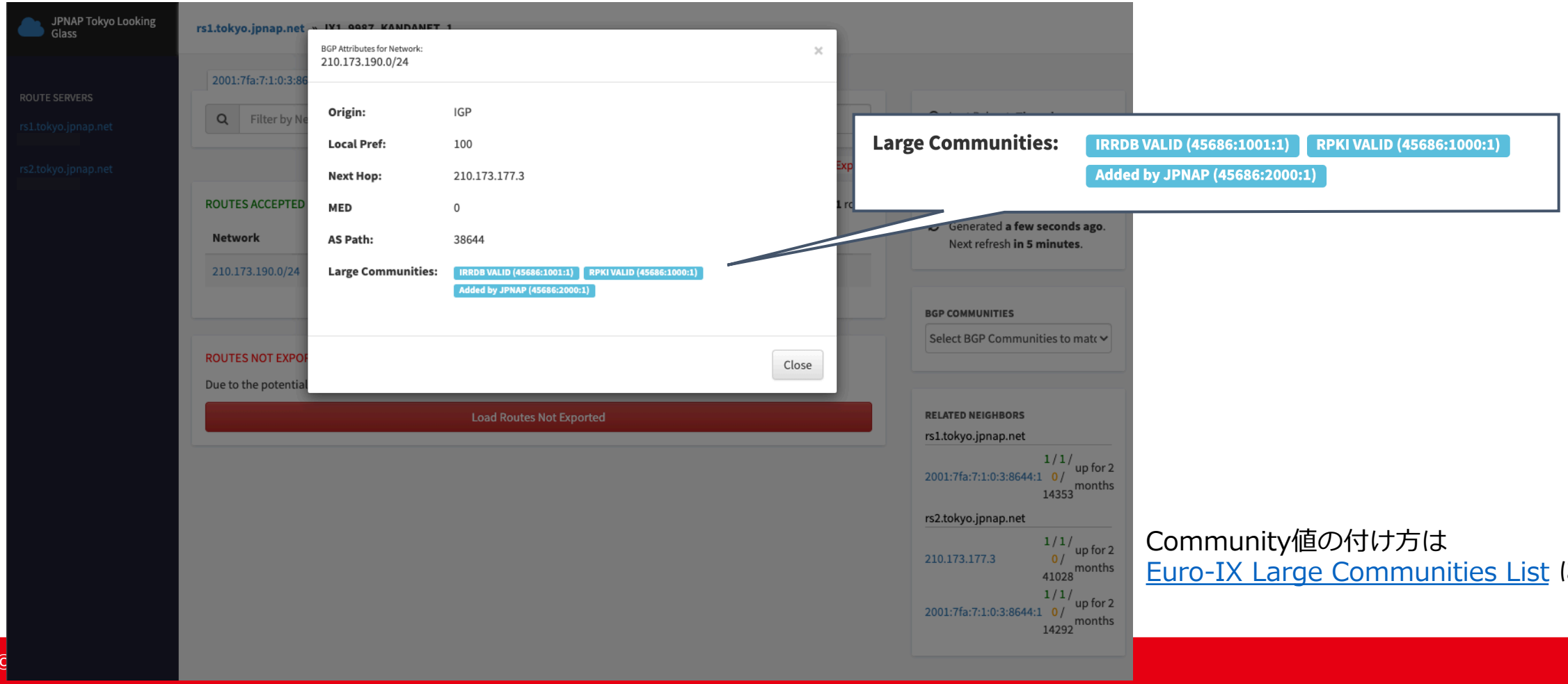
4.運用
監視やオペレーション、ユーザー向けツール

運用: 監視/オペレーション

- 監視
  - 機能監視
    - (一般的なサーバー状態の監視に+)
    - BGPd~キャッシュサーバー間のRTRセッション
    - Relying Party~RPKIリポジトリ間のrsync終了ステータス異常 (RRDPはこれから)
    - 今後検討: BGPdでユーザー経路の判定結果推移 (Valid, NotFound → Invalid) を監視
      - BIRD2.0がネイティブでmetricsを出せる訳ではないのでexporterなど要開発
      - 単純に数でしきい値を決めるのは難しく、経路単位での結果トレースはさらに一工夫必要
      - 議論: この監視結果をIXが通知してくれたら嬉しいですか？
  - 外形監視 (未実施)
    - RSに疑似的なユーザーを接続し経路の受信状況やROV状況をモニタリング？
- オペレーション
  - 手動でのVRP更新、ROV re-validation
  - 特定のASに対する一時的なROVフィルタリング無効化
    - 緊急回避策として用意はしてある
    - SLURMまでは準備していない (あったほうがいい...?)

運用: ユーザー向けツール

- OSSの Alice-LG を利用しLooking Glassを提供
- 経路に付与された (Large) BGP Community に意味づけをして表示することが可能
  - = ユーザーがRSのROV結果を確認できる



Community値の付け方は
Euro-IX Large Communities List に準拠

導入に向けたステップ

1.設計

ポリシー決め、技術選定、パラメーター検討

2.検証

設計した技術の要素検証、特にROVやRelying Partyとの接続など

3.導入

本番環境への適用に至るまでのタイムライン、顧客周知や導入結果の確認など

4.運用

監視やオペレーション、インターネット全体におけるROV導入の課題など

2020年12月24日

JPNAP RouteFEEDサービスにおいて、全てのルートサーバーでRPKI (Resource Public Key Infrastructure) による経路の広報元検証 (ROV: Route Origin Validation) を導入しました。 これによりルートサーバーを介したお客様間の経路交換がこれまで以上にセキュアになります。JPNAPは引き続きインターネットルーティングセキュリティの向上に努めてまいります。

RPKIについては JPNIC様の解説ページ をご覧ください。

https://www.mfeed.ad.jp/ でのアナウンス

問題共有: ROV普及後に発生する通信断事象

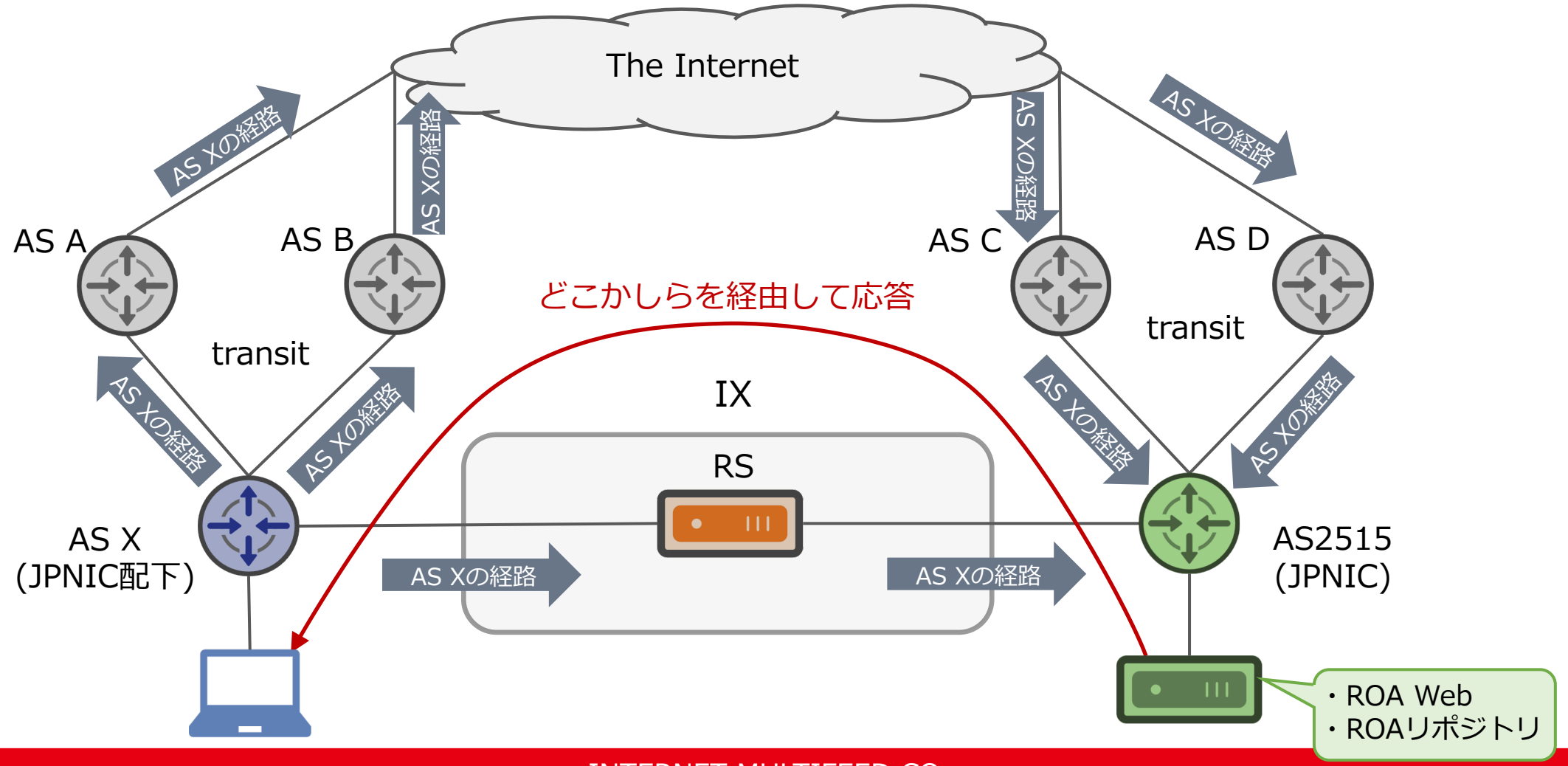
- 最後にROVが十分に普及したインターネットにおいて発生し得る問題について共有

- 背景
  - JPNAPでROV導入をユーザーにアナウンスした際、JPNIC様よりコメントをいただき議論を行う
  - 発生するのはまだ少し先であり、ある種極端な状況ではあるが、コミュニティ全体で意識すべき内容と考え共有する

- 発生事象概要
  - 対外接続先ASがROVでInvalid経路をdropしている状況下でROAの作成を誤ると、インターネットとの接続が絶たれる
  - JPNICのROA WebにもアクセスできなくなりROAの修正を行うこともできず断が継続する

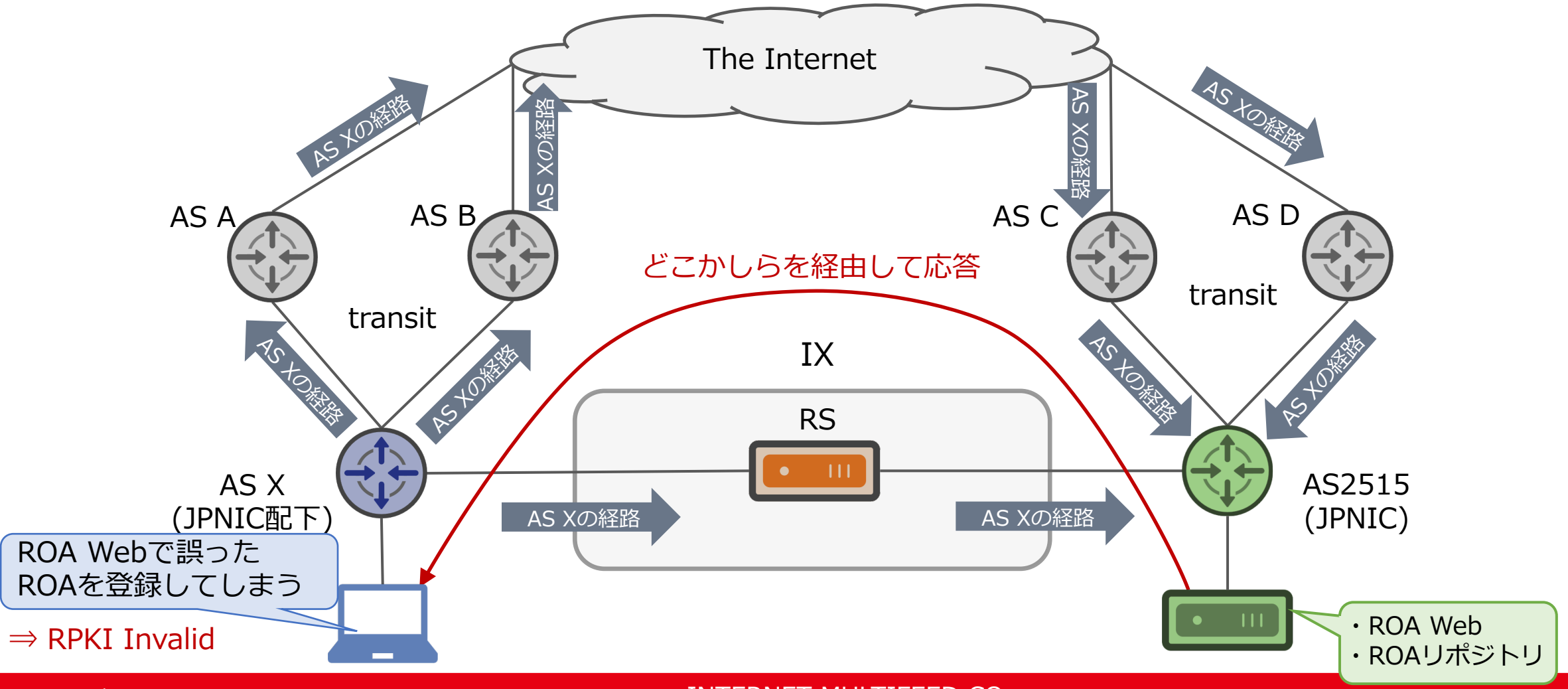
問題共有: 事象発生シナリオ

- あるAS XはJPNICからアドレスアサインを受けている
- インターネットを経由してJPNICのRPKIサービスであるROA Webにアクセスし利用する



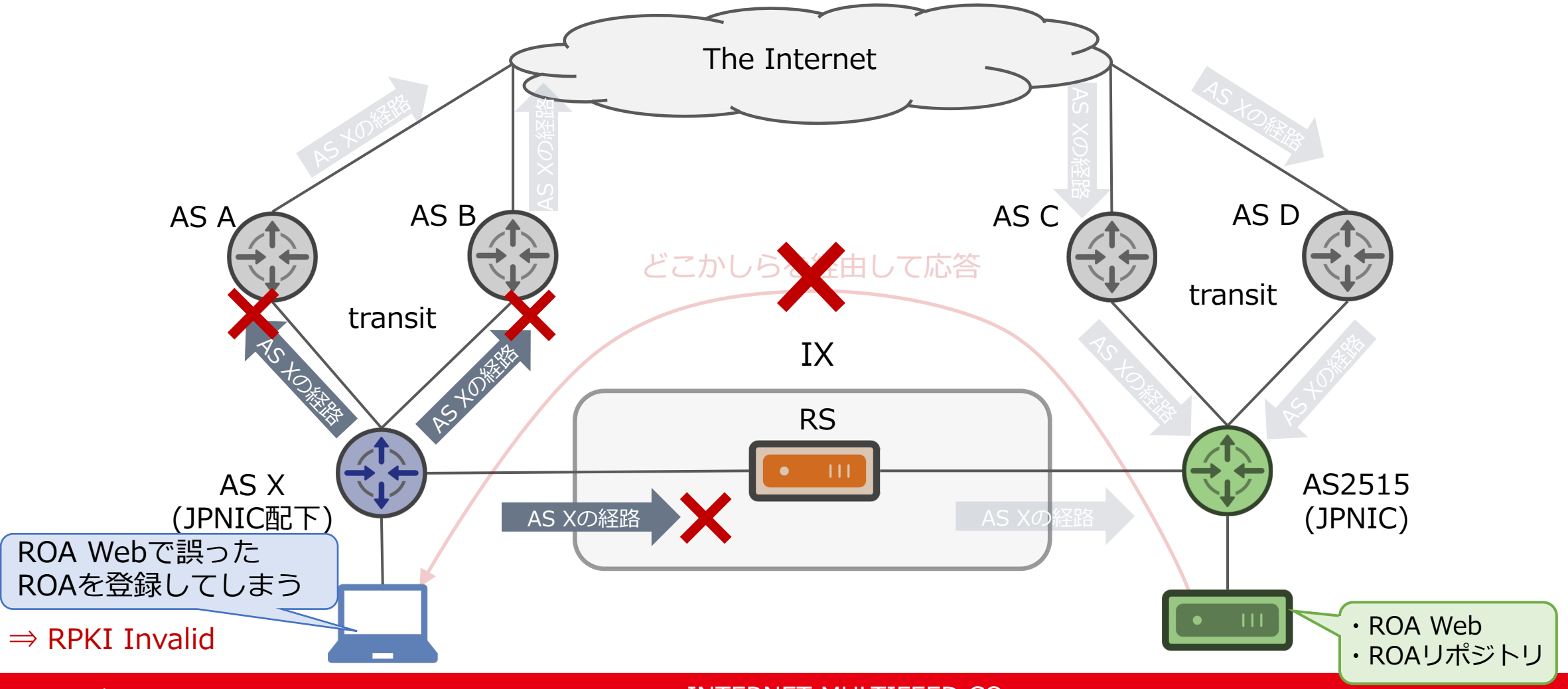
問題共有: 事象発生シナリオ

- AS Xは自社ネットワークの経路についてROA Webで誤ったROA登録をしてしまう
- 当該の経路がRPKI Invalidと判定される



問題共有: 事象発生シナリオ

- AS Xのトランジットであるas AとB、またRSはROVを実装しており、Invalidな経路をrejectする
- AS Xはインターネットから隔絶され、ROAを修正したくともROA Webにアクセスができない



問題共有: 事象の考察

- 発生条件
  - ある国内ASについて以下の3つを満たすこと
  - 1) 自身がインターネットに出るための経路についてROA登録を誤る (根本原因)
  - 2) 全てのトランジットAS、及び接続しているIXのRSがROVを実装し、Invalid経路を落とす
  - 3) JPNICとのプライベートピア、もしくはIX上でのバイラテラルピアを持たない
    - JPNICがAPNIC TALに関するROVを実施しない前提
- 備考
  - 日本国内にとどまらず他のNIR/RIRにおいても発生し得るのではないか (要調査)
  - 本質的にはRSに接続しているかは関係なく、全てのトランジットがInvalid経路を落とせば起き得る

- 事象の本質
  - インターネット上でのトラフィック交換に直接影響を与える制御をインターネットを介して行うこと
    - ネットワークサービスをインラインで制御することに等しい
    - 従来のASやIPの申請は制御ではなく、直接インターネットルーティングに影響がなかった

問題共有: 対策

- 指針: 常に何らかの方法でJPNIC ROA Webを使えるようにしておく

- AS運用者の対策
  - 対外接続先 (トランジット、IX) のROV導入状況を把握しておく
  - ROA登録/編集時はとても慎重に様子を見ながら実施する
    - JPNICのROA Webは作成したROAがすぐAPNICに反映されるよう仕様変更済み
  - 自社ネットワークに関係ない通信経路 (テザリング等) でROA Webにアクセスできるようにする
    - 無線接続する端末に資源申請者証明書をインストールしておく ※セキュリティリスクとのトレードオフ

- AS運用者+JPNICの対策
  - 自ASとJPNIC間でバイラテラル/プライベートピアを張る
    - JPNICはOpen Policy

議論したいこと

- RPKIを導入する視点で
  - 設計～運用までの各プロセスについて、持てる知見を是非共有ください
  - これから検討する、今まさに検討中の方が課題だと思う点、など

- IX利用者の視点で
  - RPKI Invalid経路を受信したいケースがあるか？
    - 実験や検証用途でInvalid経路を受信したい、等
  - こんな付加サービスがあったら使いたい？
    - ROA/経路監視サービス (経路がInvalid判定に変わったら通知)
    - IX上でのRelying Partyの提供
  - 要望や改善案

- ROAを作成/運用する視点で
  - 先の事例に対し備えていることがあれば教えてください

- その他質問やコメントなど