ProVerifによるBGPセキュリティ機構の形式的安全性検証

2025.9.4

Authors 神谷海登 三重野武彦 岡崎裕之 鈴木彦文 荒井研一 布田裕一

目次

- 本研究の目的
- 実施した内容
- BGPとは
- BGPの防御機構
- BGPの形式化
- ROVの形式化
- 検証結果
- 考察、まとめ
- 今後の展望

本研究の目的

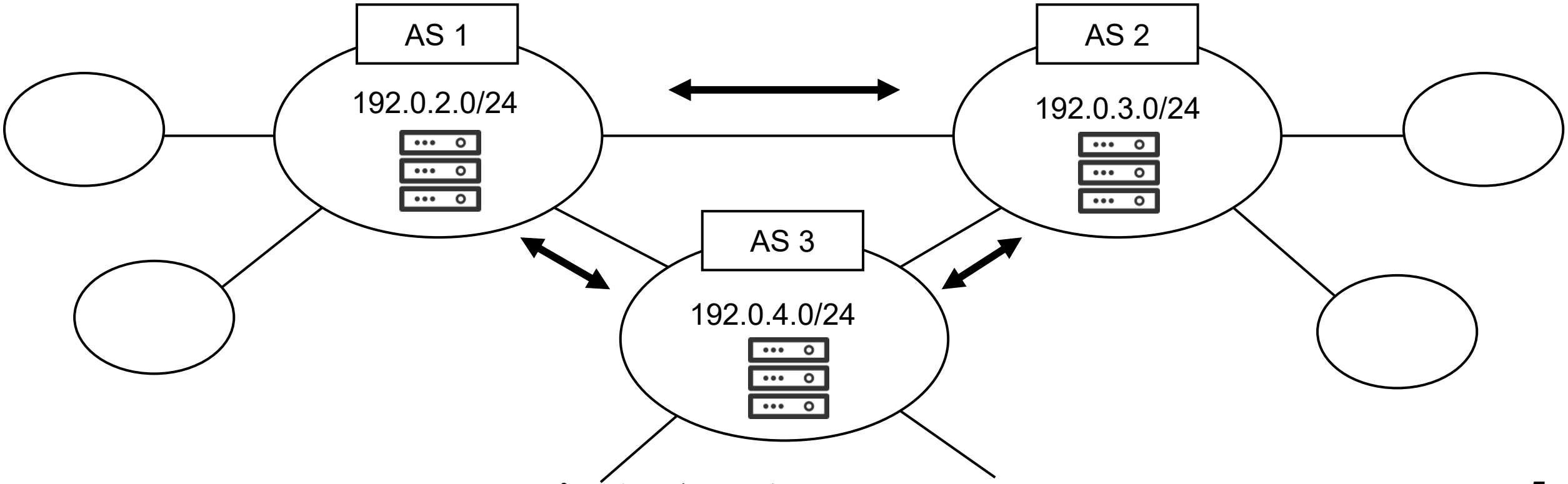
- BGPの既存防御機構の安全保証レベルの明確化と信頼性の向上
- 未知攻撃パターンの発見、対策の立案

今回実施した内容

- BGP単体のモデルと防御機構の一つであるROVを適用したモデルを作成
- それぞれのモデルにおいて主要な３つの経路ハイジャックの到達可能性をProVerifで評価
- 検証結果の比較をすることにより、ROVの有効性を評価

BGPとは

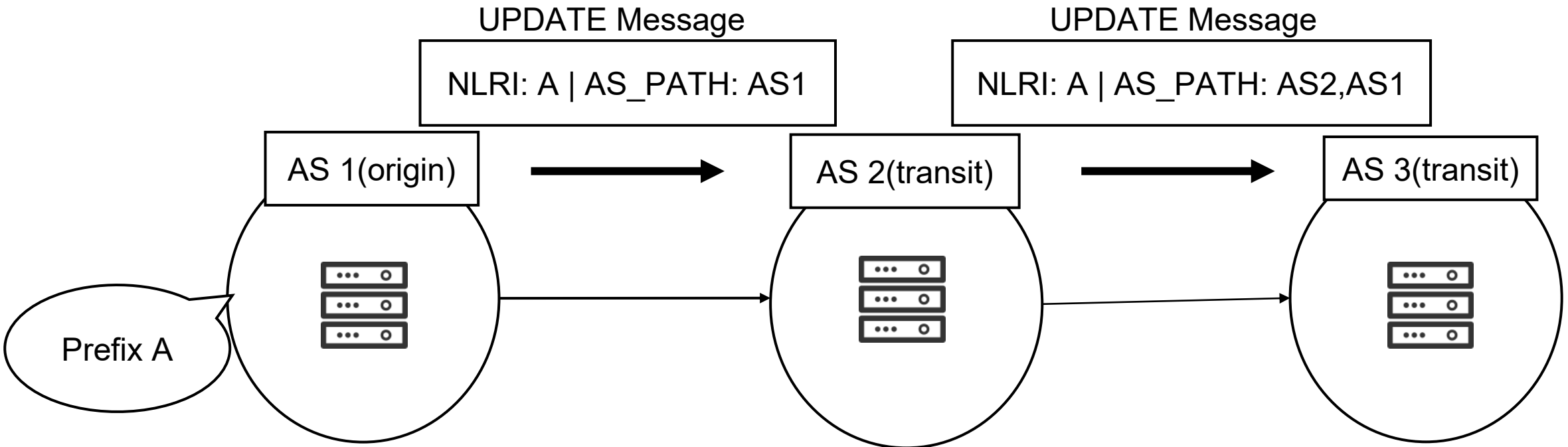
- インターネット上の自律システム(AS)間のルーティングプロトコル
- 他のBGPシステムとネットワーク到達可能性情報を交換する



※ASは企業、大学、インターネットプロバイダなどが管理するネットワーク

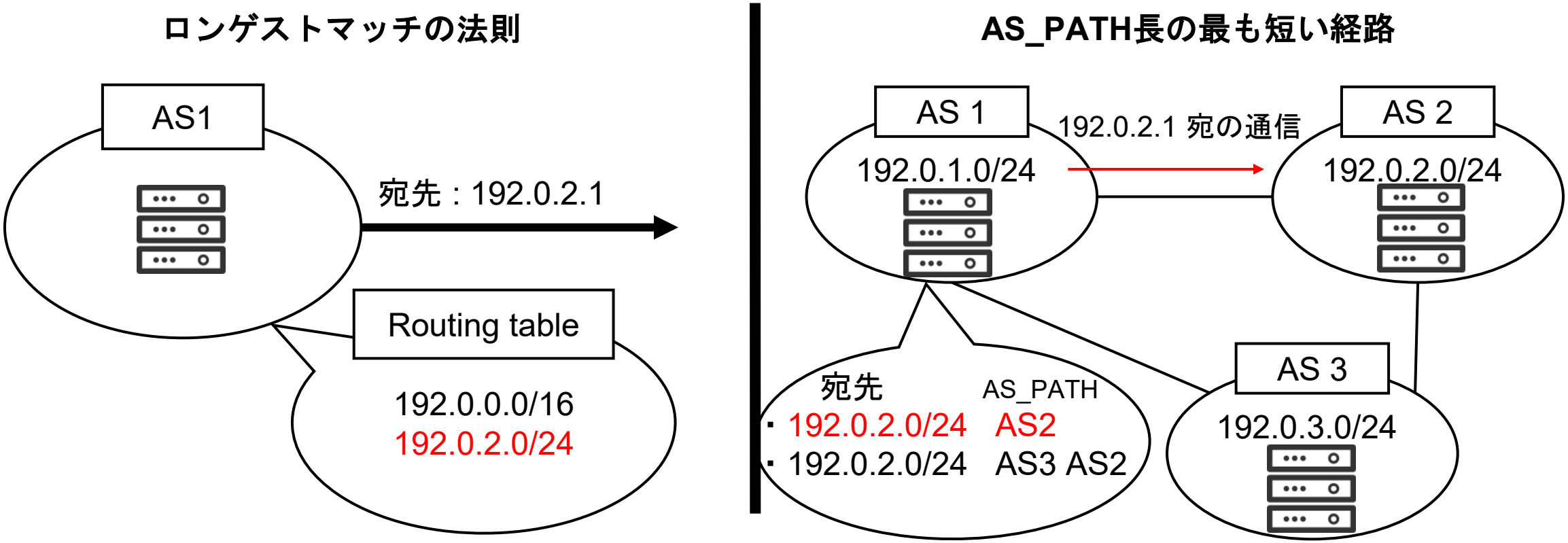
BGPの経路広告

- UPDATE Messageで到達可能性情報を交換する
  - NLRI : 広告する宛先プレフィックスのリスト
  - AS_PATH : 経路がどのASを経由して広告されたかのAS番号のリスト



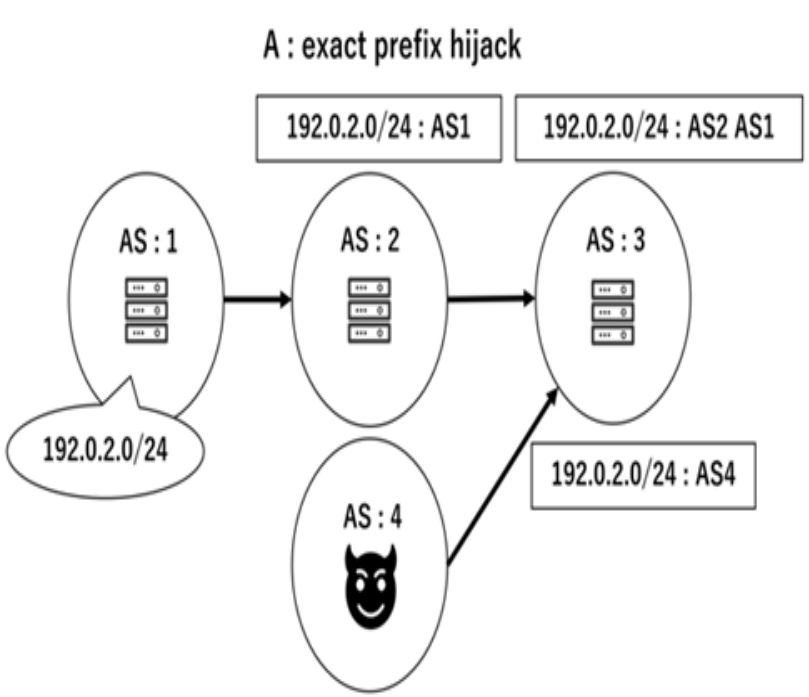
BGPの経路選択

- ロンゲストマッチの法則
- AS_PATH長の最も短い経路

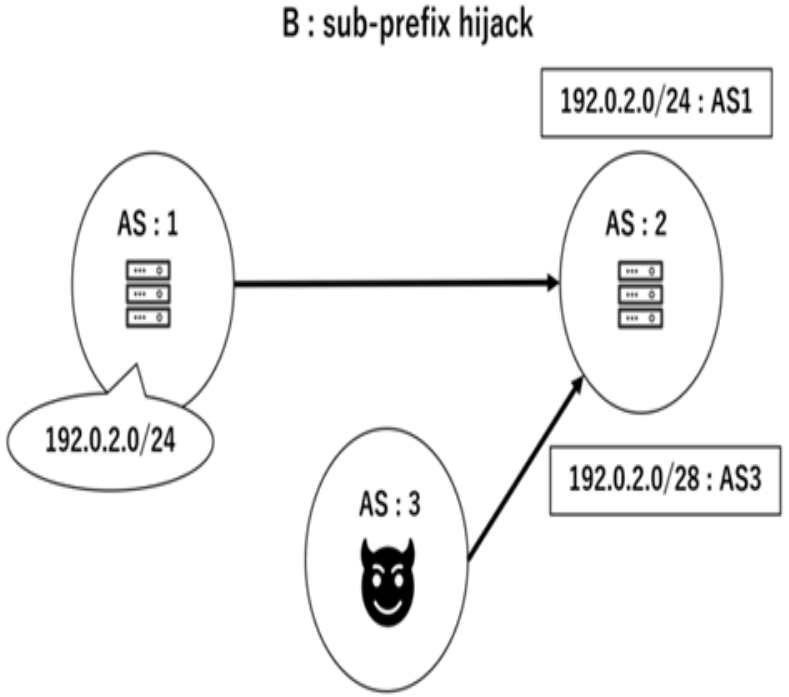


BGPの脆弱性

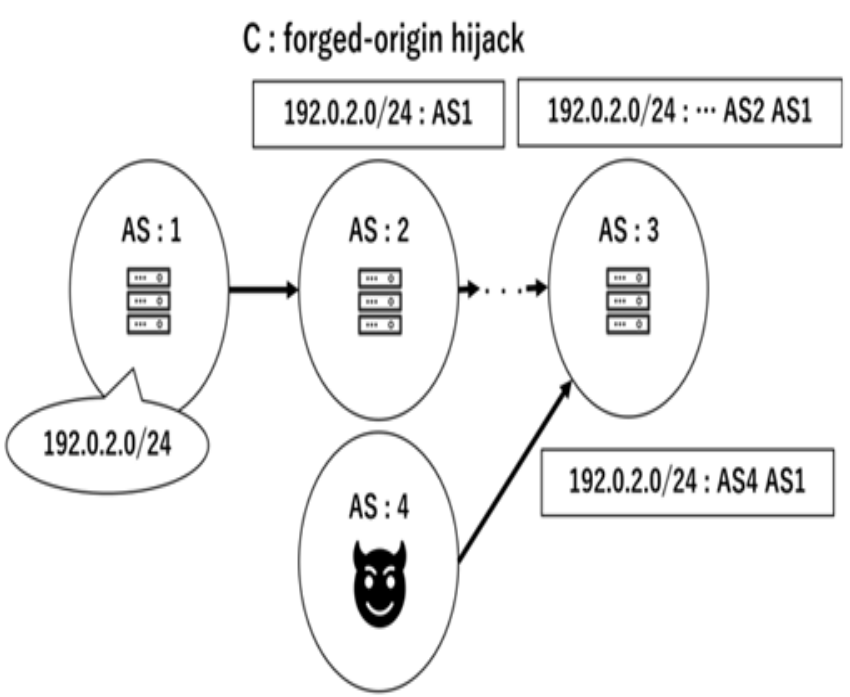
到達可能性情報を改ざんして攻撃者のASにトラフィックを誘導できてしまう



Origin ASを示すAS_PATHの末端を攻撃者自身のAS番号に書き換えて経路を広告する。



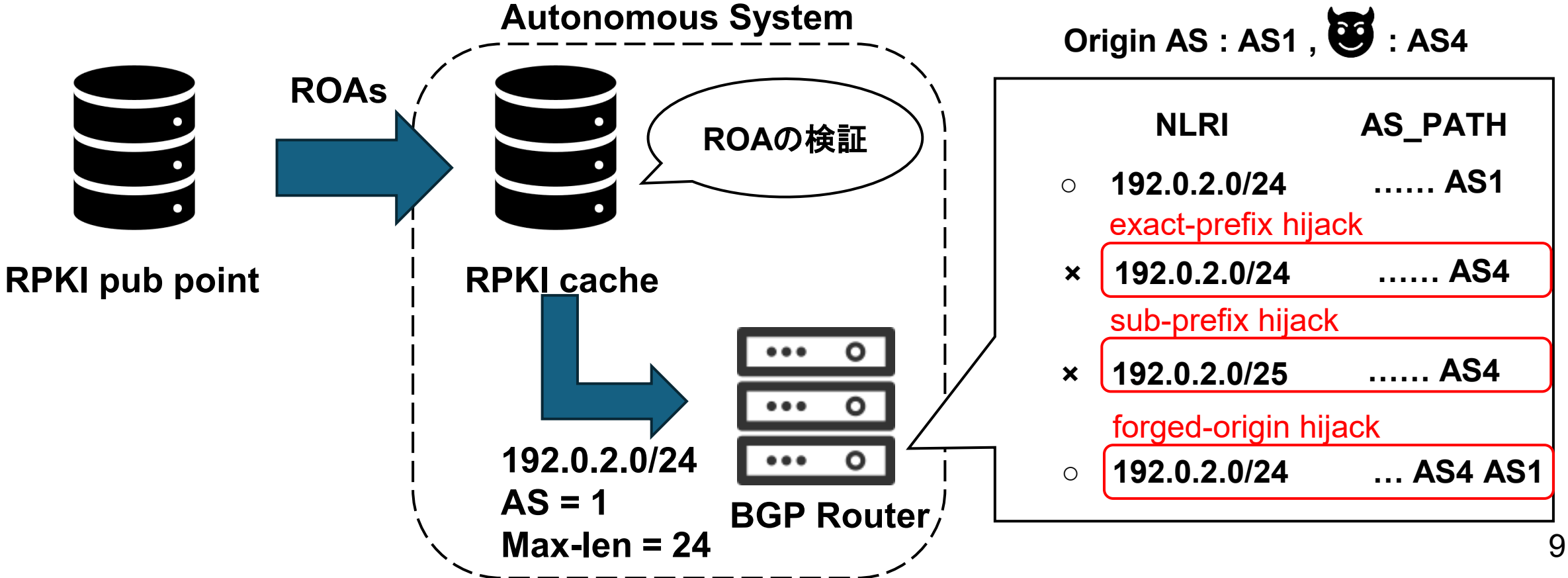
攻撃者ASがprefix長の長いより具体的なプレフィックスを広告する。



AS_PATHの末端以外に攻撃者のAS番号を挿入したAS_PATHを偽造して広告する。

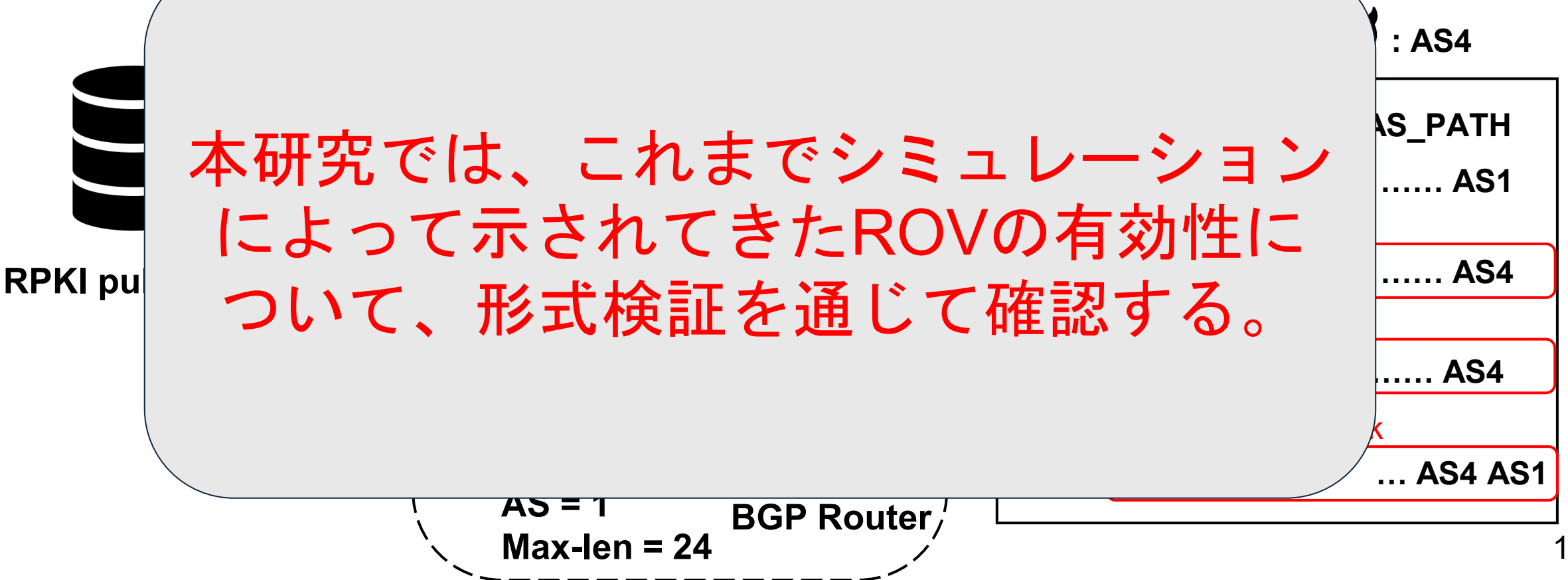
BGPの防御機構(ROV)

ROA：どのAS（Autonomous System）が特定のIPプレフィックスをBGPで広告してよいかを証明する電子署名付きの記録。
ROV：ROAによりBGPルータが受け取った経路広告が正しい送信元ASから来ているかを確認する



BGPの防御機構(ROV)

ROA : どのAS（Autonomous System）が特定のIPプレフィックスをBGPで広告してよいかを証明する電子署名付きの記録。
ROV : ROAによりBGPルータが受け取った経路広告が正しい送信元ASから来ているかを確認する



本研究では、これまでシミュレーションによって示されてきたROVの有効性について、形式検証を通じて確認する。

BGPのモデル化

経路広告

- 検証のため、Origin ASは攻撃者が書き換え可能なパブリックチャネルと書き換え不可能なプライベートチャネルの双方にUPDATEメッセージを送信する
- Transit ASは経路選択およびAS_PATHの更新を行い、次のTransit ASにUPDATEメッセージを送信する

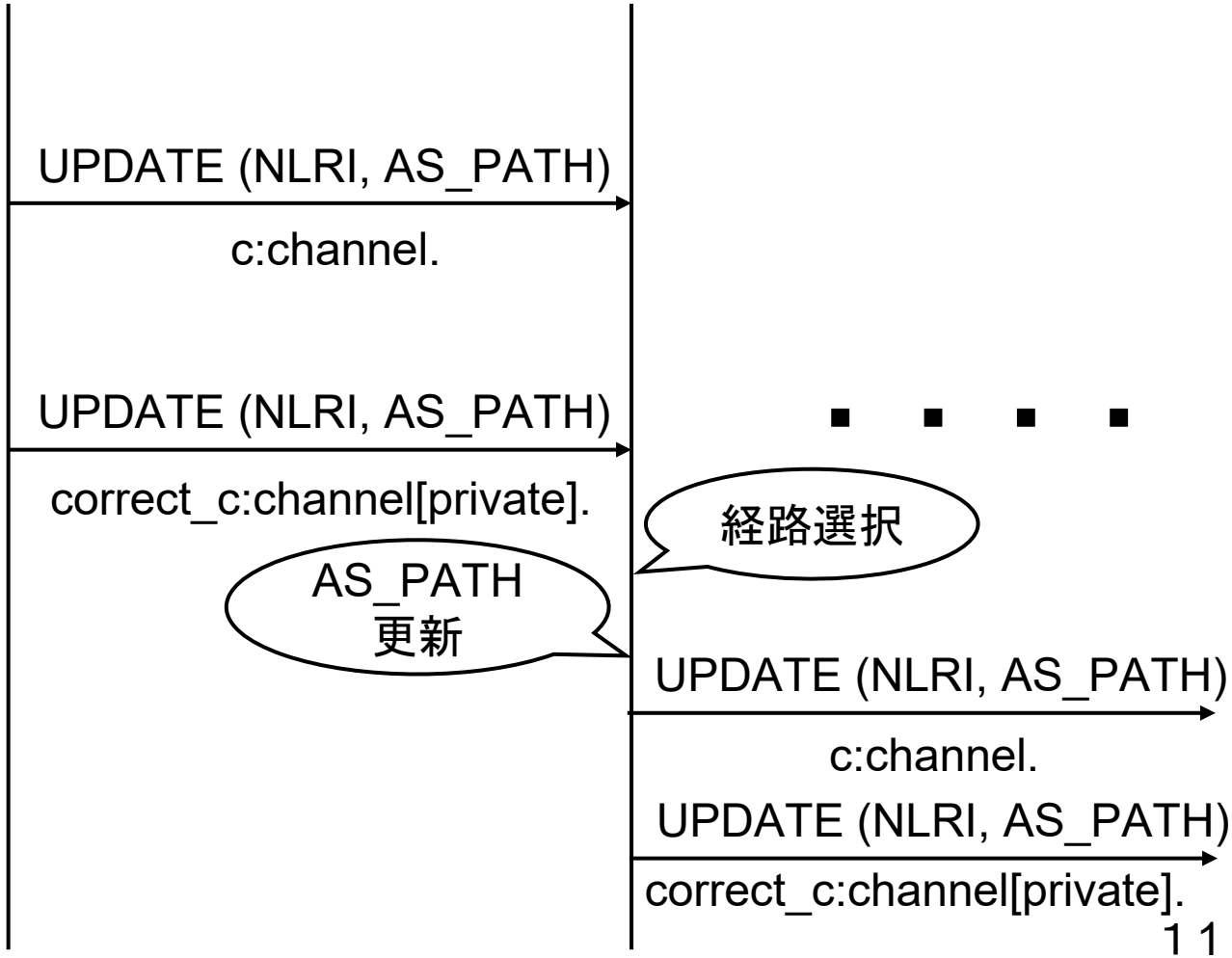
UPDATEメッセージの構成

- NLRI : (IP prefix : PREFIX型, prefix長 : nat型)

```
fun gen_nlri(PREFIX, nat): NLRI.
```

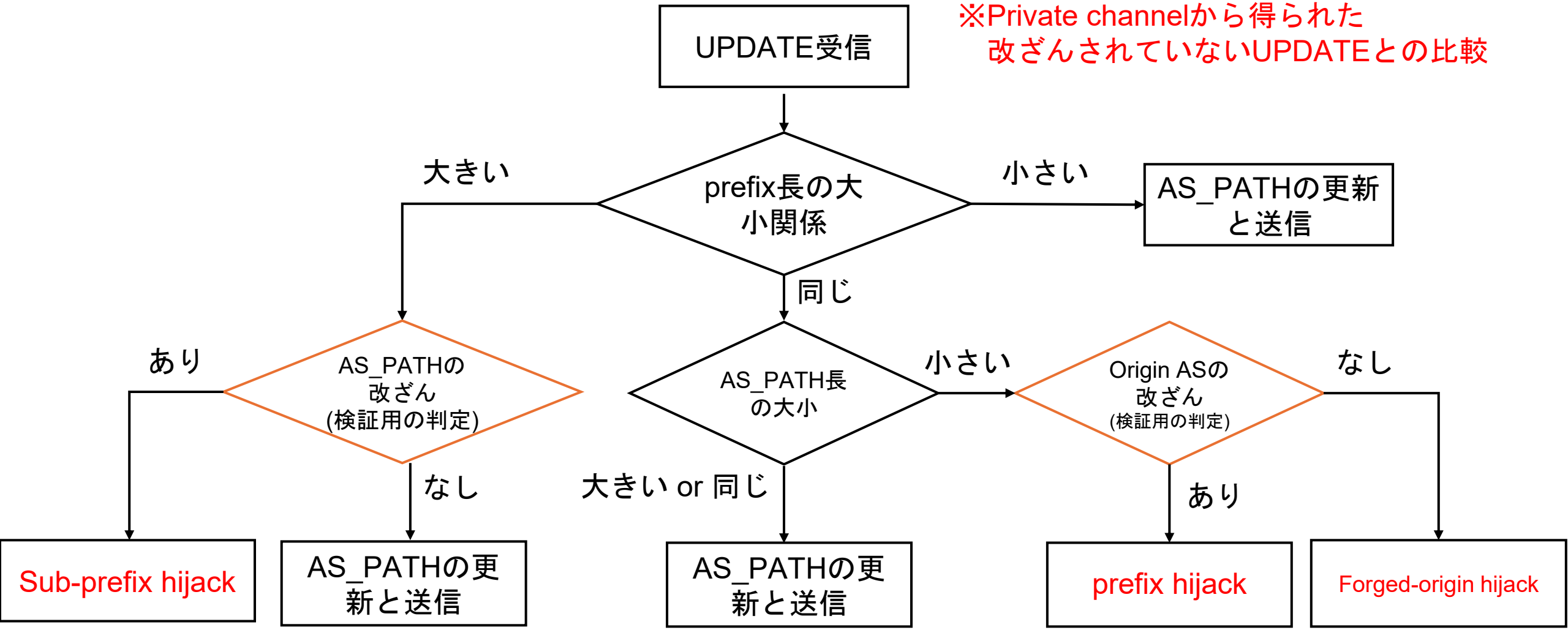
- AS_PATH : ([data:AS型])

```
const nilp : AS_PATH [data].
fun   cons(AS, AS_PATH): AS_PATH [data].
```



BGPモデルの経路選択フロー図

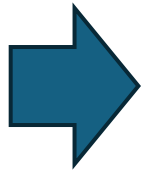
※Private channelから得られた
改ざんされていないUPDATEとの比較



ProVerifの記述

- **AS_PATH長の大小比較**

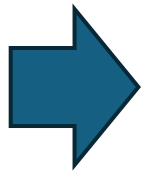
```
(* originp(p, og): 末尾（起点）AS *)
pred originp(AS_PATH, AS).
clauses
  forall a:AS;                        originp(cons(a, nilp), a);
  forall a:AS, p:AS_PATH, og:AS;      originp(p, og) -> originp(cons(a, p), og).
```



```
(* path : public channelから受信したAS_PATH*)
(* c_path : private channelから受信したAS_PATH*)
if longer(path, c_path) then          ← 長い経路を受信
else if longer(c_path, path) then     ← 短い経路を受信
else                                  ← 同じ経路を受信
```

- **Origin ASの比較**

```
(* longer(p, q): p の方が q より「厳密に」長い *)
pred longer(AS_PATH, AS_PATH).
clauses
  forall a:AS, p:AS_PATH;                    longer(cons(a,p), nilp);
  forall a1:AS, a2:AS, p1:AS_PATH, p2:AS_PATH;  longer(p1,p2) -> longer(cons(a1,p1), cons(a2,p2)).
```



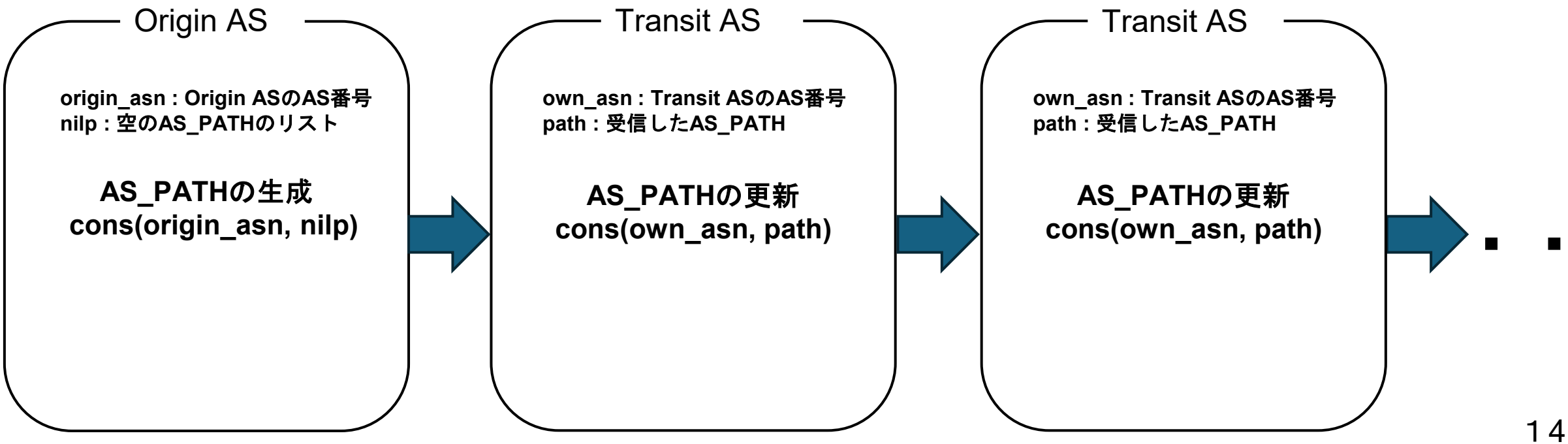
```
(* path : public channelから受信したAS_PATH*)
(* c_path : private channelから受信したAS_PATH*)
let og1:AS suchthat originp(path,   og1) in
let og2:AS suchthat originp(c_path, og2) in
if og1 = og2 then   ← Origin ASが一致
else                ← Origin ASが不一致
```

AS_PATHの更新

```
(*** AS_PATH: 純粋データのリスト ***)
const nilp : AS_PATH [data].
fun   cons(AS, AS_PATH): AS_PATH [data].
```

空のリスト

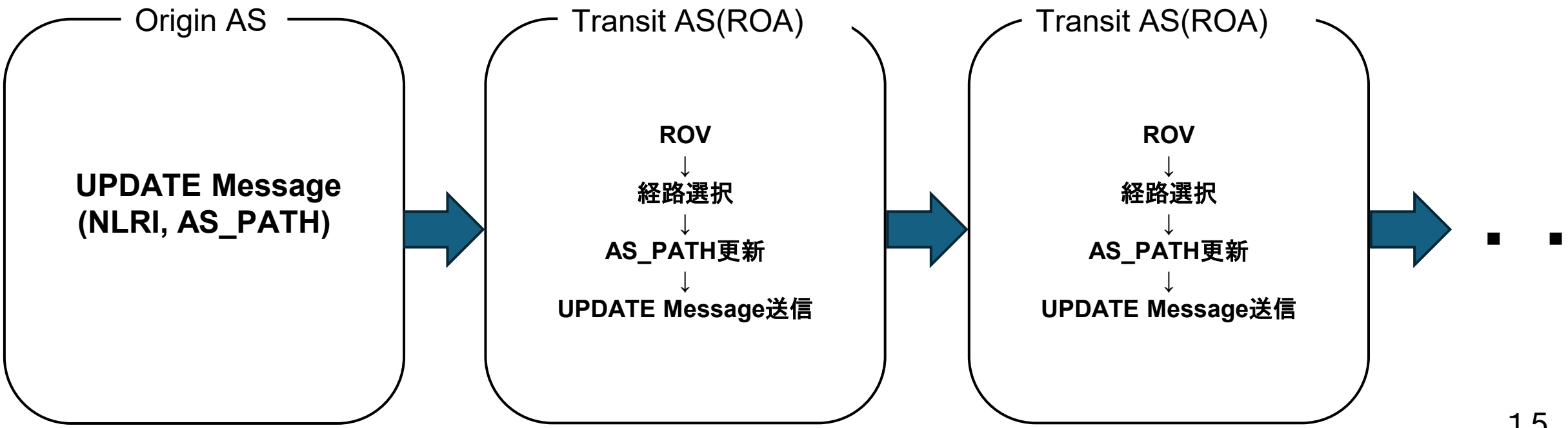
リストにASを追加(AS_PATHの更新)



ROVのモデル化

- BGPの形式化と同様、Origin ASとTransit ASのプロセスを定義し、Transit ASプロセスの引数にROAの情報を渡す
- ROA(roa_pr: ip プレフィックス, roa_og: オリジンAS, roa_prlen: プレフィックス長)

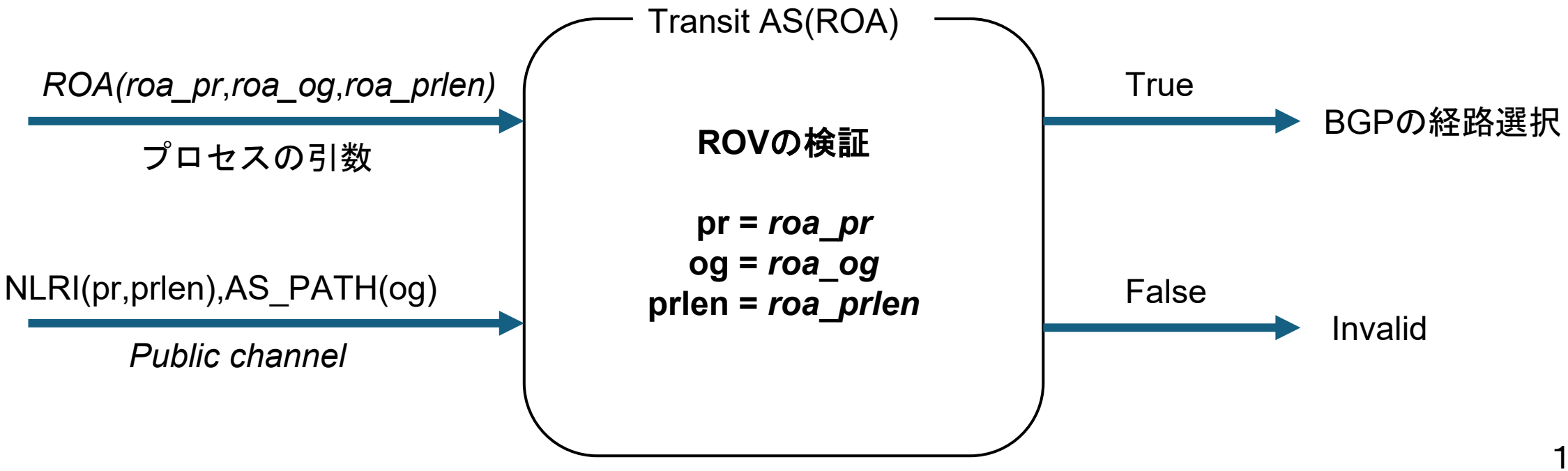
  let TransitAS(*roa_pr*:PREFIX,*roa_og*:AS,*roa_prlen*:nat)



ROV適用

- **ROV**

Public channelから受信したUPDATEメッセージ中のIPプレフィックス、プレフィックス長、オリジン ASが一致するか検証



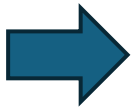
検証

query	シミュレーションによる結果
(* prefix hijack *) event succ_ph. query event(succ_ph).	BGP単体 : 攻撃成功　ROV適用 : 攻撃失敗
(* sub-prefix hijack *) event succ_sph. query event(succ_sph).	BGP単体 : 攻撃成功　ROV適用 : 攻撃失敗
(* forged-origin hijack *) event succ_foh. query event(succ_foh).	BGP単体 : 攻撃成功　ROV適用 : 攻撃成功

参考：Furuness J. *et al.* (2025). *Securing BGP ASAP: ASPA and other Post-ROV Defenses*.

BGP単体の検証結果

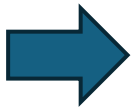
Query not event(succ_ph) is false.



```
40. The message (new_sess_1,gen_nlri(prefix_AS1[],16),cons(own_asn_1,cons(origin_asn[],nilp))) that may be sent on channel correct_c[] by 26 may be received at input {13}.
The message (new_sess_1,gen_nlri(prefix_AS1[],16),cons(og1_1,nilp)) that the attacker may have by 34 may be received at input {14}.
By 36, longer(cons(own_asn_1,cons(origin_asn[],nilp)),cons(og1_1,nilp)) is true at {27}.   ← より短いAS_PATH長の改ざんされた経路を受信
By 37, originp(cons(og1_1,nilp),og1_1) is true at {28}.   ← 改ざんされたAS_PATHのOrigin AS(og1_1)
By 39, originp(cons(own_asn_1,cons(origin_asn[],nilp)),origin_asn[]) is true at {29}.   ← 正規AS_PATHのOrigin AS(origin_asn[]))
We have og1_1 ≠ origin_asn[] && is_nat(16).   ← 正規PATHのOrigin ASと改ざんされたPATHのOrigin AS が異なる
So event succ_ph may be executed at {32}.
event(succ_ph).

41. By 40, event(succ_ph).
The goal is reached, represented in the following fact:   ← 攻撃経路あり
event(succ_ph).
```

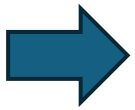
Query not event(succ_sph) is false.



```
10. The message (sess[],gen_nlri(prefix_AS1[],16),cons(origin_asn[],nilp)) that may be sent on channel correct_c[] by 1 may be received at input {13}.
The message (sess[],gen_nlri(prefix_AS1[],prlen),path_1) that the attacker may have by 9 may be received at input {14}.
We have prlen ≥ 17.   ← 正規IP Prefixのprefix長(16)よりも具体的なUPDATEを受信
So event succ_sph may be executed at {18}.
event(succ_sph).

11. By 10, event(succ_sph).
The goal is reached, represented in the following fact:   ← 攻撃経路あり
event(succ_sph).
```

Query not event(succ_foh) is false.



```
40. The message (new_sess_1,gen_nlri(prefix_AS1[],16),cons(own_asn_1,cons(origin_asn[],nilp))) that may be sent on channel correct_c[] by 26 may be received at input {13}.
The message (new_sess_1,gen_nlri(prefix_AS1[],16),cons(origin_asn[],nilp)) that the attacker may have by 35 may be received at input {14}.
By 37, longer(cons(own_asn_1,cons(origin_asn[],nilp)),cons(origin_asn[],nilp)) is true at {27}.   ← より短いAS_PATH長の改ざんされた経路を受信
By 38, originp(cons(origin_asn[],nilp),origin_asn[]) is true at {28}.   ← 改ざんされたAS_PATHのOrigin AS(oigin_asn[])
By 39, originp(cons(own_asn_1,cons(origin_asn[],nilp)),origin_asn[]) is true at {29}.   ← 正規AS_PATHのOrigin AS(origin_asn[])
We have is_nat(16).
So event succ_foh may be executed at {31}.
event(succ_foh).

41. By 40, event(succ_foh).
The goal is reached, represented in the following fact:   ← 攻撃経路あり
event(succ_foh).
```

ROV適用時の検証結果

Query not event(succ_ph) is true. → 到達不可(攻撃失敗)

Query not event(succ_sph) is true. → 到達不可(攻撃失敗)

Query not event(succ_foh) cannot be proved. →

```
23. The message (new_sess_1,gen_nlri(prefix_AS1[],16),cons(own_asn_1,cons(origin_asn[],nilp))) that may be sent on channel correct_c[] by 9 may be received at input {16}.
The message (new_sess_1,gen_nlri(prefix_AS1[],16),cons(origin_asn[],nilp)) that the attacker may have by 19 may be received at input {17}.
By 8, originp(cons(origin_asn[],nilp),origin_asn[]) is true at {19}.   ← 改ざんされたAS_PATHのOrigin AS(origin_asn[])
By 20, originp(cons(own_asn_1,cons(origin_asn[],nilp)),origin_asn[]) is true at {20}.   ← 正規AS_PATHのOrigin AS(origin_asn)
By 22, longer(cons(own_asn_1,cons(origin_asn[],nilp)),cons(origin_asn[],nilp)) is true at {33}.   ← より短いAS_PATH長の改ざんされた経路を受信
So event succ_foh may be executed at {35}.
event(succ_foh).

24. By 23, event(succ_foh).
The goal is reached, represented in the following fact:   ← 攻撃経路あり
event(succ_foh).
```

まとめ

• BGPの脆弱性確認

BGP単体では prefix hijack・sub-prefix hijack・forged-origin hijack の3種類の攻撃が成立することを形式手法で確認した。

• ROVの効果の検証

ROVを適用したモデルでは、prefix hijackとsub-prefix hijackは到達不可能（防御可能）であることを確認した。一方で、forged-origin hijackについては完全に防ぐことができず、依然として攻撃成功の可能性が残る結果となった。

本研究は、これまでシミュレーションで行っていたBGP防御機構の評価を、形式的手法により安全性を検証する試みである。検証モデルにおいても、既存の研究と同じ結論を導けたことは、形式検証の妥当性と有効性を示す。

限界と今後の課題

- **現実とのギャップ**

ROV導入率を考慮した検証は未実施

- **モデル化の制約**

ASポリシー・多様なトポロジー・Valley-Free Routingを再現困難

- **手法の補完**

形式検証は理想的な状況の検証が可能だが複雑なモデルの検証は困難であるため、シミュレーションによる検証も必要

- **今後の展望**

ASPAなどの追加セキュリティ機構やROVとの併用効果の形式検証

ご清聴ありがとうございました