PPASPA: Aumentando a Segurança do Roteamento da Internet com Autorizações de Provedores por Prefixo

Sthefany C. Duquini[1], Pedro de B. Marcos[2], Ítalo Cunha[3], Ronaldo A. Ferreira[1]

[1]Universidade Federal de Mato Grosso do Sul (UFMS)
[2]Universidade Federal do Rio Grande (FURG)
[3]Universidade Federal de Minas Gerais (UFMG)

sthefany.duquini@ufms.br, pbmarcos@furg.br, cunha@dcc.ufmg.br
raf@facom.ufms.br

Abstract. *ASPA (Autonomous System Provider Authorization), a recent mechanism for Internet route validation, allows an autonomous system (AS) to publish signed objects specifying which providers are authorized to announce its prefixes. However, because authorizations are defined only at the AS level, the mechanism becomes inflexible in scenarios involving multiple providers and distinct prefix policies. This work presents PPASPA, an extension to ASPA that enables prefix-specific authorizations, thereby covering practical situations not covered by the original model. We implement PPASPA in the BIRD routing software, preserving ASPA's original AS-path validation logic while modifying only the mechanism used to identify authorized providers for prefix-specific policies. An evaluation using synthetic and trace-inspired scenarios shows that the computational overhead introduced by PPASPA is low, which may enable and facilitate its wide adoption on the Internet.*

Resumo. *O ASPA (Autonomous System Provider Authorization) é um mecanismo recente para validação de rotas na Internet que permite um sistema autônomo (AS) publicar objetos assinados indicando quais provedores estão autorizados a anunciar seus prefixos. Entretanto, como as autorizações são definidas por AS, o mecanismo é pouco flexível em cenários com múltiplos provedores e políticas distintas por prefixo. Este trabalho apresenta PPASPA, uma extensão de ASPA que permite associar autorizações específicas a prefixos e que amplia a proteção para situações práticas não contempladas no modelo original. PPASPA foi implementado no software de roteamento BIRD, preservando a lógica de validação de AS-path do ASPA e modificando apenas o mecanismo de identificação dos provedores autorizados a anunciar um prefixo com política específica. A avaliação com dados sintéticos e dados inspirados em traces reais mostra que os custos computacionais introduzidos por PPASPA são baixos, o que pode viabilizar e facilitar sua ampla adoção na Internet.*

1. Introdução

O *Border Gateway Protocol* (BGP) é o protocolo responsável pelo roteamento interdomínio da Internet e permite a troca de informações de alcançabilidade entre Sistemas Autônomos (ASes) [Rekhter et al. 2006]. Entretanto, o BGP foi concebido sem mecanismos de segurança nativos [Butler et al. 2004], baseando-se essencialmente na confiança

entre ASes. Essa limitação torna o protocolo vulnerável a ataques como o sequestro de prefixos (*prefix hijack*), em que um AS anuncia prefixos que não lhe pertencem, o que pode causar interrupção de serviços, desvio de tráfego e, consequentemente, prejuízos financeiros [Butler et al. 2010, Rodday et al. 2024].

Para mitigar esse problema, a IETF (*Internet Engineering Task Force*) padronizou a RPKI (*Resource Public Key Infrastructure*) [Bush and Austein 2017], uma infraestrutura de chaves públicas que permite aos detentores de recursos IP publicar objetos assinados digitalmente, chamados ROA (*Route Origin Authorization*), que certificam que um AS está autorizado a anunciar um conjunto de prefixos. Com base nesses objetos, roteadores podem validar um anúncio de rotas via ROV (*Route Origin Validation*) e rejeitar ou reduzir a prioridade de anúncios inválidos. A adoção de ROV tem crescido continuamente e representa um avanço significativo para a segurança do roteamento interdomínio [Testart et al. 2024, Rodday et al. 2024].

Embora eficaz na prevenção de simples sequestros de prefixo, ROV é incapaz de evitar vazamentos de rota e pode ser contornado por um atacante determinado. Um agente malicioso pode forjar um anúncio, inserindo artificialmente o número do AS (ASN) de origem legítimo no início do AS-path, para contornar a validação ROV e sequestrar um prefixo [Azimov et al. 2025, Furuness et al. 2025]. Tais ataques pós-ROV mostram que a validação de origem não é suficiente para garantir a integridade do roteamento.

ASPA (*Autonomous System Provider Authorization*) foi proposto recentemente como um mecanismo de validação de caminhos BGP [Azimov et al. 2024, Azimov et al. 2025]. ASPA permite que um AS publique, de forma criptograficamente assinada, a lista de provedores de trânsito que estão autorizados a receber seus anúncios de rota. Com essas informações, é possível verificar se a sequência de ASes em um AS-path observado em uma rota é consistente com as relações comerciais declaradas, o que oferece proteção contra vazamentos de rota e inserção do AS de origem no início do AS-path [Azimov et al. 2025, Furuness et al. 2025].

No ASPA, as autorizações são definidas globalmente por AS, sem distinção entre prefixos; ou seja, um AS publica um objeto ASPA com a lista dos ASes que são seus provedores. Embora esse modelo seja suficiente em muitos casos, ele é restritivo em cenários nos quais um AS adota políticas de trânsito distintas para diferentes prefixos, como em *multihoming*, anúncios seletivos, trânsito parcial [Giotsas et al. 2014], engenharia de tráfego [Kastanakis et al. 2023] ou mitigação de ataques DDoS por *scrubbing* [Khadka 2025, Khadka et al. 2026]. Nessas situações, apenas parte dos prefixos pode ser propagada por determinados provedores, o que é impossível declarar em objetos ASPA. Uma abordagem que permita autorizações específicas por prefixo pode, portanto, aumentar a expressividade e a segurança do mecanismo.

Este trabalho propõe PPASPA (*Per-Prefix ASPA*) e sua implementação no software de roteamento BIRD 2 [CZ.NIC 2026] para suportar autorizações ASPA associadas a prefixos específicos e proteger anúncios de rota em situações práticas não contempladas em ASPA. PPASPA preserva a lógica original de ASPA e adiciona o mecanismo de seleção da regra aplicável dependendo do prefixo sendo anunciado. Na ausência de regras específicas por prefixo, uma regra global com a mesma semântica de ASPA tradicional é utilizada como *fallback*. Assim, PPASPA adiciona granularidade às autorizações, mas

mantém compatibilidade com o modelo global já previsto para ASPA.

Experimentos extensivos de desempenho com dados sintéticos e com parâmetros derivados de traces reais de mensagens BGP (*updates*) coletadas pelo projeto Route Views [University of Oregon 2026] mostram que os custos computacionais introduzidos por PPASPA são baixos, o que pode viabilizar sua adoção na Internet. O código-fonte de PPASPA e os dados e os scripts utilizados na avaliação estão disponíveis em https://github.com/ppaspa.

As principais contribuições deste trabalho são: *(i)* proposta de um modelo ASPA com granularidade por prefixo; *(ii)* implementação de suporte a regras ASPA prefixadas no BIRD; *(iii)* desenvolvimento de três estratégias de seleção da regra prefixada, baseadas em busca linear, *trie* e uma política híbrida; e *(iv)* avaliação experimental dos custos introduzidos pela extensão em cenários sintéticos controlados e em cenários parametrizados.

2. Fundamentação Teórica

Esta seção apresenta brevemente conceitos fundamentais para a compreensão do PPASPA.

2.1. BGP e Seleção de Rotas

A troca de informações no protocolo BGP ocorre por meio de mensagens do tipo *update*, que informam aos vizinhos quando novas rotas ficam disponíveis (*announcement*) ou quando rotas anunciadas anteriormente ficam indisponíveis (*withdraw*). Quando múltiplas rotas para um prefixo estão disponíveis, roteadores BGP executam o algoritmo de seleção de rotas, que compara os seguintes atributos em ordem:

- **LocalPref:** O *local preference* indica o grau de preferência de uma rota e é configurado pelo operador da rede. Em geral, rotas recebidas de clientes têm LocalPref maior do que rotas recebidas de parceiros, e rotas de parceiros têm LocalPref maior do que rotas recebidas de provedores.
- **Comprimento do AS-path:** Número de ASes atravessados pela rota do AS atual até a origem. Rotas menores são mais preferidas que rotas longas.
- **Critérios de desempate intradomínio:** O desempate entre rotas com o mesmo LocalPref e comprimento do AS-path continua utilizando métricas intradomínio, como o valor do atributo MED (*Multi-Exit Discriminator*) recebido de redes vizinhas, seguido do custo da rota intradomínio até o roteador de borda.
- **Outros critérios de desempate:** Em caso de empate nos critérios intradomínio, o BGP utiliza uma série de critérios de desempate, como o endereço IP do próximo roteador (*next-hop*) ou o *timestamp* de recebimento da rota.

2.2. Validação da Origem de Rotas

Um objeto ROA (*Route Origin Authorization*) é um certificado criptograficamente assinado que autoriza um AS x a anunciar um prefixo IP p e, opcionalmente, subprefixos de p até um limite máximo de comprimento de prefixo definido pelo atributo maxlength. A validação ROV (*Route Origin Validation*) é o processo de utilizar os objetos ROAs para verificar se uma rota foi originada por uma rede autorizada. O ROV verifica apenas o AS de origem e ignora o restante do AS-path, o que motivou novos mecanismos de validação da rota completa, como BGPsec [Lepinski and Sriram 2017, Borchert et al. 2021], em que todo AS no AS-path deve assinar o anúncio, e ASPA [Azimov et al. 2025, Furuness et al. 2025].

2.3. *Autonomous System Provider Authorization* (ASPA)

O ASPA é uma alternativa mais leve ao BGPsec e atualmente em processo de padronização na IETF [Azimov et al. 2025]. Em vez de autenticar criptograficamente cada salto do AS-path, o ASPA foca na verificação da plausibilidade do caminho. Um AS cliente (*Customer AS*, CAS) pode criar um objeto ASPA declarando seu conjunto de ASes provedores (*Provider ASes*, PAS) de trânsito. Com base nessas relações cliente-provedor, a validação por ASPA consegue detectar anúncios com AS-paths inválidos, como vazamento de rotas ou anúncios com origem forjada para burlar ROV.

Algoritmo de Validação de Caminho. O algoritmo de verificação de AS-paths no ASPA verifica se uma rota viola a lógica *valley-free* de roteamento interdomínio [Gao and Rexford 2001]. Mais especificamente, o algoritmo verifica se o AS-path, a partir da origem, atravessa uma sequência potencialmente vazia de enlaces cliente-provedor (também chamada de rampa de subida, *up-ramp*), zero ou um enlace de parceria par-a-par, e uma sequência potencialmente vazia de enlaces provedor-cliente (também chamada de rampa de descida, *down-ramp*). Um AS-path que não segue essa sequência viola a política *valley-free* e é declarado inválido.

Os enlaces cliente-provedor e provedor-cliente são identificados a partir dos objetos ASPA criados por ASes que declaram seus provedores na RPKI. O algoritmo de verificação de ASPA pode ser implantado incrementalmente na Internet e, por isso, considera que ASes podem não ter adotado ASPA nem declarado seus provedores. Um sistema autônomo que não declara seus provedores introduz incertezas no processo de verificação de ASPA. Em particular, quando um enlace A–B é atravessado e ambas as redes não declararam seus provedores, é impossível saber se a relação é cliente-provedor, par-a-par ou provedor-cliente. Consequentemente, a validação de ASPA pode indicar que o estado de verificação de uma rota é desconhecido quando não há informações suficientes.

2.4. BIRD Routing Daemon

O BIRD [CZ.NIC 2026] é um *daemon* de roteamento de código aberto amplamente utilizado na Internet, especialmente em IXPs (*Internet Exchange Points*). Sua arquitetura é organizada em torno de protocolos, canais, filtros e tabelas internas. Protocolos produzem ou consomem rotas e objetos auxiliares; canais ligam protocolos e tabelas; filtros aplicam políticas locais; e tabelas mantêm os dados processados pelo núcleo do sistema.

A Figura 1 ilustra uma arquitetura simplificada do BIRD, mas suficiente para os propósitos deste trabalho. No fluxo principal, rotas recebidas por protocolos como BGP e *static* (*i.e.,* rotas estáticas configuradas pelo operador de rede) são trocadas através de canais de comunicação, que aplicam filtros de exportação e importação. Objetos auxiliares de validação, como objetos ROA e ASPA, podem ser configurados explicitamente pelo protocolo *static* ou carregados por meio do protocolo RPKI e indexados em tabelas específicas para estes objetos. As bases de ROA e ASPA são consultadas por funções específicas chamadas pelos filtros de rotas configurados nos canais entre protocolos.

Os detalhes internos de indexação e armazenamento do BIRD relevantes para este trabalho são retomados na Seção 4, onde descrevemos as modificações realizadas.

Protocolo *static* e Injeção de Objetos. O protocolo *static* permite injetar manualmente rotas e objetos auxiliares por meio de um arquivo de configuração. Neste trabalho, ele foi

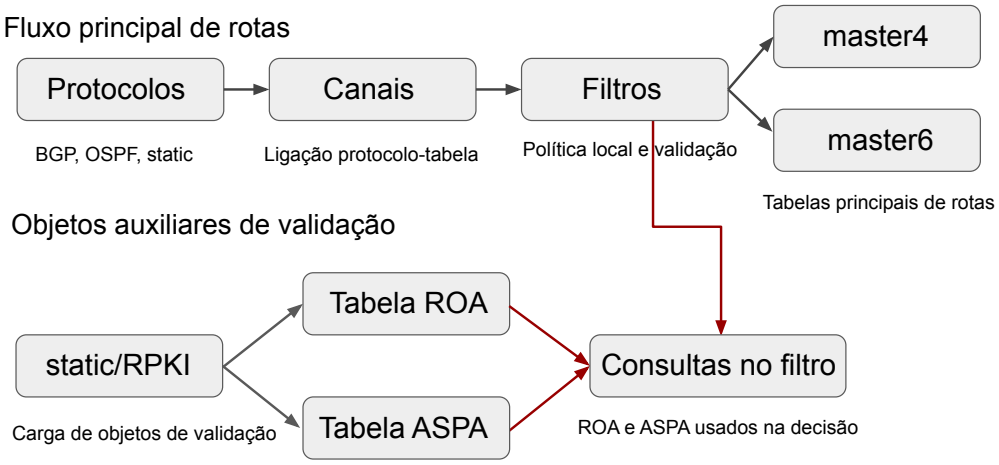

Figura 1. Fluxo simplificado do BIRD, ilustrando apenas os módulos mais relevantes para este trabalho.

utilizado tanto para inserir rotas de teste quanto para carregar objetos ASPA utilizados na validação, o que permite controlar explicitamente a quantidade de rotas, o tamanho dos AS_PATHs, o número de regras por AS e a presença de regras específicas por prefixo.

3. Exemplos de Limitações do ASPA

No ASPA, um AS cliente autoriza todos seus provedores a anunciarem todos os seus prefixos. Esta autorização ampla impede a proteção adequada de prefixos em diversos cenários práticos. Por exemplo, um cliente de serviços de mitigação DDoS por *scrubbing* sob demanda [Khadka 2025, Khadka et al. 2026] pode determinar que apenas parte de seus prefixos seja submetida ao serviço de *scrubbing* e eventualmente redirecionados ao provedor de mitigação. Entretanto, como o provedor de *scrubbing* precisa aparecer na autorização global ASPA do AS cliente, um anúncio indevido de outro prefixo continua compatível com a política publicada. Em caso de erro de configuração, abuso ou comprometimento do relacionamento entre cliente e provedor de *scrubbing*, o ASPA tradicional não distingue esse anúncio de um uso legítimo da autorização.

Um problema similar aparece em cenários de anúncio seletivo e de relações híbridas [Giotsas et al. 2014, Jin et al. 2019]. Para permitir a propagação legítima de apenas parte dos prefixos por um provedor secundário, o ASPA tradicional precisa incluí-lo na autorização global da origem, o que o autoriza para todos os prefixos do AS do cliente. Assim, o provedor secundário pode anunciar um prefixo que não faz parte do acordo, e o anúncio continua compatível com a autorização publicada.

No PPASPA, que apresentamos a seguir, o AS cliente pode restringir a autorização de anúncios de prefixos a provedores específicos, como provedores de serviço de *scrubbing*, bem como provedores de trânsito parcial. Os exemplos desta seção motivam diretamente a extensão proposta na Seção 4.

4. PPASPA: Autorizações de Provedores por Prefixo

O modelo atual de ASPA associa a cada AS cliente uma autorização única para um conjunto de provedores de trânsito. Esse modelo assume que a relação entre provedores de trânsito e clientes é sempre a mesma para todos os prefixos anunciados. De forma

ilustrativa, uma autorização ASPA tradicional para um AS65001 de exemplo pode ser representada como: {customer = 65001, providers = [65010, 65020]}. No PPASPA, um objeto pode conter uma lista global de provedores e múltiplas regras prefixadas refinando a lista de provedores para subconjuntos específicos do espaço de endereçamento, por exemplo: {customer = 65001, providers = [65010, 65020], prefixRules = {203.0.113.0/24 = [65010], 198.51.132.0/22-24 = [65020]}}.

4.1. Extensão Proposta ao ASPA

O PPASPA segue os *drafts* atuais do ASPA em padronização na IETF, preservando integralmente a semântica original de validação de ASPA e modificando apenas o mecanismo de seleção da autorização aplicável ao enlace entre o AS de origem e seu provedor direto.

Mais precisamente, um AS a pode publicar um objeto PPASPA contendo uma regra global com um conjunto de provedores e múltiplas regras prefixadas. Cada regra prefixada associa um prefixo p, e opcionalmente subprefixos até o comprimento máximo c, a um conjunto específico de provedores autorizados. Durante a validação, consideramos a regra prefixada mais específica que cobre o prefixo anunciado. Caso nenhuma regra específica exista, consideramos a regra global. Para os demais enlaces do AS-path, apenas as regras globais continuam sendo consideradas.

Essa decisão limita o custo adicional da validação por prefixo ao primeiro enlace do caminho e preserva o comportamento original do ASPA para os ASes intermediários. Além disso, a abordagem concentra a granularidade adicional justamente na origem, a entidade mais apta a definir políticas específicas para seus próprios prefixos.

Considerações operacionais. A maior granularidade introduzida pelo PPASPA aumenta o volume de informação a ser administrado em comparação ao ASPA tradicional. Porém, a adoção do PPASPA é opcional e retrocompatível com o ASPA tradicional. Além disso, mecanismos como o RPSL já são utilizados por operadores de rede para descrever políticas por vizinho e por prefixo, incluindo regras de importação, exportação e filtros sobre conjuntos de prefixos [Rekhter et al. 1999, Blunk et al. 2005]. Nesse sentido, a granularidade introduzida por PPASPA não é estranha a operadores de rede e é comum na especificação de políticas de roteamento.

Implementação. Nossa implementação do PPASPA no BIRD modifica a representação interna dos objetos ASPA e a lógica de seleção da regra aplicável. Estendemos a estrutura ASPA original para suportar regras prefixadas contendo prefixo IP, comprimento máximo de subprefixos e lista de provedores autorizados. Também modificamos o *parser* do protocolo static para permitir a definição de regras prefixadas nos arquivos de configuração.

4.2. Variantes de seleção da regra prefixada

Uma vez introduzidas múltiplas regras ASPA por origem, a eficiência da seleção da regra mais específica passa a ser um aspecto importante da implementação. Avaliamos três variantes que compartilham a mesma semântica de validação.

Na variante *linear*, as regras associadas ao AS de origem são percorridas integralmente, e a implementação mantém a regra prefixada mais específica compatível com o prefixo da rota. Na variante com *trie*, as regras prefixadas de cada origem são indexadas em uma estrutura de busca por prefixos, permitindo operações eficientes de *longest-prefix match*.

Por fim, implementamos uma variante *híbrida* que combina as duas abordagens. Inicialmente, a busca ocorre de forma linear. Entretanto, origens com quantidade elevada de regras prefixadas e consultas frequentes podem ser promovidas dinamicamente para uma estrutura baseada em *trie*. Na implementação atual, essa promoção só é considerada para origens com pelo menos 16 regras prefixadas. Além disso, a decisão depende de um escore de atividade recente dado pelo produto entre o número de regras prefixadas da origem e um contador de consultas recentes. Esse contador é incrementado a cada consulta prefixada e sofre decaimento ao longo do tempo. Mais especificamente, quando 32 ou mais consultas são observadas pela estrutura híbrida sem novo acesso àquela origem, esse contador é reduzido progressivamente. A promoção ocorre quando esse escore atinge o limiar 2048. Esse processo evita que origens que foram intensamente consultadas apenas no passado permaneçam indefinidamente promovidas. A criação de *tries* adicionais aumenta o consumo de memória, como mostrado na Seção 6; por isso, o número de estruturas ativas é limitado globalmente a 256 *tries*, reservando o uso da estrutura às origens com maior escore recente. Quando esse limite é atingido, a implementação pode despromover a *trie* ativa com menor escore. O Algoritmo 1 apresenta a descrição completa do algoritmo de seleção da regra prefixada na variante híbrida.

Algorithm 1 Seleção da regra prefixada na variante híbrida

Require: AS de origem AS_o, prefixo p da rota
E ← entrada correspondente a AS_o na tabela ASPA
atualizar contador de consultas recentes e escore de E
if E.trie = null **and** E.regrasPrefixadas ≥ 16 **and** E.escore ≥ 2048 **then**
    if limite de memória excedido **then**
        liberar a *trie* ativa com menor escore, se necessário
    end if
    E.trie ← construir a *trie* sobre E.regrasPrefixadas para AS_o
end if
if E.trie $\neq$ null **then**
    regraPrefixada ← E.trie.longestPrefixMatch(p)
else
    regraPrefixada ← buscaLinear(E.regrasPrefixadas)
end if
return regraPrefixada **if** regraPrefixada $\neq$ null **else** E.regraGlobal

5. Ambiente de Avaliação

Foram realizados testes de desempenho para comparar o custo de validação ASPA entre a implementação original do BIRD e as versões modificadas para PPASPA. As avaliações visam medir tanto o impacto global no processo do BIRD quanto o custo interno da rotina de validação ASPA. Os testes foram organizados em duas frentes complementares: cenários sintéticos, voltados ao isolamento de variáveis estruturais, e cenários parametrizados com base em métricas extraídas de arquivos de *updates* do projeto Route Views.

5.1. Infraestrutura

O ambiente experimental foi organizado em quatro árvores do BIRD 2, são elas: BIRD com ASPA original, BIRD com PPASPA linear (sem *trie*), BIRD com PPASPA com *trie* e BIRD com variante híbrida. Cada ambiente possui seu próprio contêiner de execução. A instrumentação foi adicionada apenas para coleta de métricas, sem alterar o desenho central dos testes ou o núcleo do BIRD. A Figura 2 ilustra o fluxo no BIRD modificado, com marcações em vermelho nos módulos alterados.

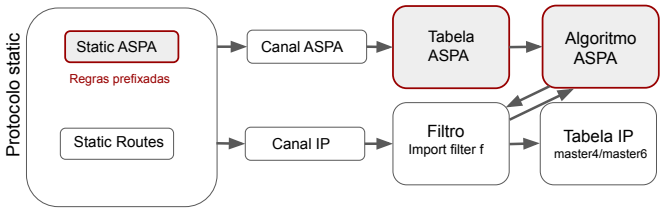

Figura 2. BIRD modificado com os módulos alterados marcados em vermelho.

Para automatizar os experimentos, foi desenvolvido um orquestrador em Python. Ele gera, para cada ponto da campanha, os arquivos de configuração .conf correspondentes a cada variante avaliada, preservando o mesmo cenário experimental e ajustando apenas as diferenças necessárias entre o ASPA original e as variantes prefixadas. Em seguida, o orquestrador executa os testes nos contêineres, coleta métricas internas do BIRD e consolida os resultados em arquivos CSV para análise posterior.

Em todas as variantes, as regras ASPA e PPASPA e as rotas de teste contidas nos arquivos .conf são injetadas por meio do protocolo static do BIRD. Cada rota gerada pelo orquestrador recebe um bgp_path explicitamente construído de acordo com o cenário experimental. A importação de rotas passa pelo filtro que invoca a função aspa_check(). Comparamos o PPASPA com o ASPA original, utilizado como base de comparação. Cada uma das variantes (linear, *trie* e híbrida) é avaliada em três cenários distintos:

- **Regra Presente (`hit`):** cenário em que uma regra prefixada é encontrada e utilizada.
- **Regra Ausente (`miss`):** cenário em que nenhuma regra prefixada é encontrada e a validação recorre à regra global por *fallback*.
- **Nenhuma Regra (`global`):** cenário em que nenhuma regra prefixada é definida.

5.2. Cenários Sintéticos e Cenários Parametrizados com Métricas do Route Views

Os experimentos foram divididos em dois grupos. O primeiro corresponde a cenários sintéticos controlados, nos quais o objetivo principal é observar, de forma isolada, o impacto de cada parâmetro sobre o custo de validação. Nesses cenários, parâmetros como diversidade de origem, comprimento do AS-path, quantidade de provedores autorizados por regra e número de regras prefixadas são variados de forma controlada, de modo a permitir a análise separada de seus efeitos. Assim, quando nos referimos, por exemplo, a cenários de variação da diversidade de origem, tratam-se de configurações em que a proporção de ASes de origem distintos é gradualmente alterada, com o objetivo de identificar como essa variação afeta o comportamento da modificação proposta.

O segundo grupo corresponde a cenários sintéticos cujos parâmetros foram definidos a partir de medições obtidas de arquivos de atualização de rotas (*dumps*) do RouteViews. As distribuições da quantidade de anúncios por origem, do número de prefixos por

origem e do comprimento dos AS-paths são geradas sinteticamente a partir da distribuição de observações extraídas do RouteViews. Neste grupo, apesar de as rotas serem geradas sinteticamente, elas são representativas de cenários inspirados na realidade.

5.3. Parâmetros Variáveis

Neste trabalho, **diversidade de origem** indica quantos ASes de origem distintos aparecem no conjunto de rotas gerado. Considere um experimento com N rotas no total, isto é, com `routes` = N, e um valor para `origin_diversity_pct`. Se `origin_diversity_pct` = 0, todas as rotas pertencem a uma única origem. Caso contrário, o número de origens distintas é calculado como $\lceil N \cdot$ `origin_diversity_pct`$/100 \rceil$. Esse parâmetro afeta diretamente a cardinalidade de origens observadas no experimento e, consequentemente, o custo de reaproveitamento ou reconstrução das estruturas utilizadas na validação ASPA e PPASPA. Em particular, diversidade de origem igual a 0% representa a concentração máxima das rotas em uma única origem, enquanto valores mais altos distribuem as rotas entre um número maior de ASes de origem.

Nos cenários sintéticos controlados, foram utilizados os seguintes parâmetros:

- `routes`: número de rotas injetadas no BIRD, fixado em 10.000 nos cenários sintéticos controlados e em 1.000.000 nos cenários parametrizados a partir de coletas do RouteViews;
- `path_len`: comprimento dos AS-paths, fixado em 4 em todos os experimentos apresentados;
- `providers_per_rule`: quantidade de provedores autorizados por regra ASPA ou PPASPA, fixada em 10;
- `origin_diversity_pct`: percentual de diversidade de ASes de origem no conjunto sintético de rotas. Nos cenários sintéticos controlados em que variamos a diversidade de origem, avaliamos valores entre 0% e 100%, em passos de 10%; nos cenários sintéticos de variação do número de regras, utilizamos 0% e 50%;
- `prefix_rules`: número de regras prefixadas por AS nos cenários sintéticos controlados. Nos experimentos em que variamos a diversidade de origem, esse valor foi fixado em 64; nos experimentos de variação do número de regras, avaliamos os valores $2, 4, 8, 16, 24, 32, 48, 64, 96, 128, 192, 256, 384, 512, 768$ e 1024;
- `prefix_rules` e `origin_diversity_pct`: nos cenários parametrizados a partir de medições do RouteViews, avaliamos $2, 4, 6, 8, 10, 12, 14$ e 16 regras prefixadas por origem e valores de diversidade de origem de 1% e 2%.

O parâmetro `providers_per_rule` foi utilizado apenas para padronizar o tamanho das listas de provedores autorizados em cada regra gerada pelo orquestrador. Esse parâmetro não representa uma topologia real específica. Seu papel neste trabalho é controlar uniformemente o tamanho do conteúdo associado a cada regra ASPA ou PPASPA. Além disso, o custo de acesso à lista de provedores autorizados não constitui uma contribuição específica do PPASPA, pois a extensão proposta altera o mecanismo de seleção da regra aplicável, e não a lógica básica de verificação da lista de provedores.

Executamos cada configuração 10 vezes e apresentamos intervalos de confiança de 95%. Nos cenários sintéticos controlados, exceto nas configurações que variam a quantidade de regras prefixadas, utilizamos 64 regras prefixadas por AS. Nos cenários de

variação do número de regras prefixadas, as configurações experimentais foram combinadas com diversidade de origem de até 50%. Essa limitação foi adotada porque o aumento da diversidade de origem eleva o número de ASes de origem distintos no experimento; ao mesmo tempo, o aumento do número de regras prefixadas faz com que cada uma dessas origens passe a carregar mais regras e mais estado associado para validação. Como consequência, o volume total de regras prefixadas e das estruturas mantidas em memória cresce fortemente, formando combinações extremas de parâmetros. Cada campanha experimental é definida pelo conjunto de variantes avaliadas, pelos cenários de avaliação incluídos, pelo número de repetições medidas e pela grade de parâmetros considerada.

5.4. Métricas

Para avaliar o custo computacional de cada implementação de PPASPA, foram coletadas as seguintes métricas em cada execução: (1) *tempo total de execução*; (2) *tempo de CPU em modo usuário*, que contabiliza o tempo de CPU utilizado pelo BIRD, incluindo carregamento dos objetos de autorização e validação de rotas; (3) *tempo de CPU em modo kernel*, que contabiliza o tempo de CPU utilizado pelo Linux durante cada experimento; (4) *tempo de CPU de validação*, que contabiliza o tempo utilizado pelo BIRD na função de validação dos AS-paths; (5) *tempo médio de CPU por validação*, que corresponde à divisão do tempo de CPU de validação pelo número de validações realizadas; e (6) *pico de memória residente*, que quantifica a quantidade máxima de memória utilizada pelo BIRD.

6. Resultados

Esta seção apresenta os resultados da avaliação do custo computacional da introdução de regras PPASPA no BIRD. O objetivo é caracterizar como a granularidade por prefixo impacta o custo interno da validação de rotas e o consumo de memória à medida que variam o número de origens distintas e o número de regras prefixadas por origem. Em todos os experimentos, os valores reportados correspondem à média das execuções e aos respectivos intervalos de confiança de 95%.

Os cenários de **Regra Ausente** (*miss*), descritos na Seção 5, ficaram muito próximos dos cenários de **Regra Presente** (*hit*) e, por isso, não são mostrados nas tabelas a seguir. Nesta seção, comparamos principalmente o tempo total de leitura e importação de rotas (`wall_static`), o pico de memória residente (`rss_mb`) e o custo médio interno da validação de rotas (`avg_aspa`) entre o ASPA original e as variantes prefixadas de PPASPA. A métrica mais diretamente relacionada ao impacto da extensão proposta é `avg_aspa`, pois ela mede o custo adicional introduzido na seleção da regra aplicável. Já `rss_mb` caracteriza o custo de memória associado à manutenção das regras prefixadas, e `wall_static` permite observar como esse custo afeta o comportamento do processo como um todo.

6.1. Diversidade de Origem

O primeiro experimento varia a diversidade de ASes de origem entre 0% e 100% e utiliza 10 mil rotas, comprimento de caminho igual a 4, 10 provedores autorizados por regra e 64 regras prefixadas por origem. Esse experimento mostra como a cardinalidade do conjunto de origens afeta cada estratégia de seleção da regra prefixada. Nos casos em que nenhuma regra prefixada é utilizada, `linear-global`, `trie-global` e `hybrid-global` permaneceram próximos do baseline, o que indica que o custo adicional das modificações é desprezível quando a validação utiliza apenas a autorização global.

Tabela 1. Cenário de variação na diversidade de origens. `wall_static` e `avg_aspa` em segundos; `rss_mb` em MB.

Div	Caso	wall_static	rss_mb	avg_aspa
0%	original-global	0.0074 ± 0.0002	20.70 ± 0.09	$(1.398 \pm 0.032)\mathrm{e}{-7}$
0%	linear-hit	0.0118 ± 0.0001	20.80 ± 0.09	$(5.671 \pm 0.036)\mathrm{e}{-7}$
0%	trie-hit	0.0078 ± 0.0000	20.92 ± 0.05	$(1.845 \pm 0.010)\mathrm{e}{-7}$
0%	hybrid-hit	0.0083 ± 0.0002	20.96 ± 0.06	$(1.769 \pm 0.035)\mathrm{e}{-7}$
50%	original-global	0.0112 ± 0.0001	30.90 ± 0.09	$(3.207 \pm 0.040)\mathrm{e}{-7}$
50%	linear-hit	0.0606 ± 0.0023	265.31 ± 0.06	$(4.434 \pm 0.224)\mathrm{e}{-6}$
50%	trie-hit	0.1646 ± 0.0092	347.50 ± 0.10	$(1.460 \pm 0.088)\mathrm{e}{-5}$
50%	hybrid-hit	0.0581 ± 0.0012	267.38 ± 0.07	$(4.314 \pm 0.086)\mathrm{e}{-6}$
100%	original-global	0.0138 ± 0.0003	41.08 ± 0.06	$(4.072 \pm 0.183)\mathrm{e}{-7}$
100%	linear-hit	0.0858 ± 0.0078	509.64 ± 0.09	$(4.513 \pm 0.471)\mathrm{e}{-6}$
100%	trie-hit	0.3218 ± 0.0278	674.02 ± 0.08	$(2.746 \pm 0.247)\mathrm{e}{-5}$
100%	hybrid-hit	0.0826 ± 0.0011	513.66 ± 0.05	$(4.686 \pm 0.023)\mathrm{e}{-6}$

A Tabela 1 resume três pontos representativos desse conjunto. O comportamento mais relevante aparece nos cenários *hit*. Em 0% de diversidade de origem, em que uma única origem concentra todas as rotas, a *trie* e a híbrida apresentam custo médio de validação muito inferior ao da variante linear: `avg_aspa` cai de $\approx 5{,}7\mathrm{e}{-7}$ s em `linear-hit` para $\approx 1{,}8\mathrm{e}{-7}$ s em `trie-hit` e `hybrid-hit`. Quando o número de origens distintas cresce, a *trie* perde essa vantagem de forma acentuada. Em 50% de diversidade de origem, por exemplo, `trie-hit` sobe para 0.1646 s de `wall_static` e $1{,}46\mathrm{e}{-5}$ s em `avg_aspa`, enquanto `linear-hit` e `hybrid-hit` permanecem em 0.0606 s e 0.0581 s de `wall_static`. Em 100% de diversidade de origem, a mesma tendência aparece: a variante híbrida fica próxima da linear e muito abaixo da *trie*. Neste caso, `hybrid-hit` registra 0.0826 s de `wall_static`, contra 0.0858 s da linear e 0.3218 s da *trie*.

Esses resultados mostram que o custo computacional da granularidade por prefixo depende fortemente da diversidade de origens, mas que em todos os casos o custo da validação de uma rota é baixo. Quando poucas origens concentram muitas consultas, o custo adicional da extensão pode ser mantido baixo com estruturas orientadas a prefixo. Quando a diversidade cresce, o custo da construção das *tries* passa a dominar. Assim, a política híbrida pode atuar como mecanismo de contenção dos extremos de custo observados nas abordagens puramente linear e puramente baseada em *trie*.

6.2. Variação do Número de Regras

O segundo experimento também utiliza 10 mil rotas, mas varia o número de regras prefixadas entre 2 e 1024, e considera dois níveis de diversidade: 0% e 50%. Esse experimento separa com mais clareza o efeito da quantidade de regras por origem do efeito da cardinalidade do conjunto de origens.

A Tabela 2 resume seis combinações representativas. Com 0% de diversidade de origem, a diferença entre as estratégias mostra que a varredura linear cresce rapidamente à medida que o número de regras aumenta, pois o *lookup* passa a depender diretamente da quantidade de entradas associadas àquela origem. Isso aparece de forma mais clara em `avg_aspa`: `linear-hit` passa de $1{,}5\mathrm{e}{-7}$ s com 2 regras para $6{,}88\mathrm{e}{-6}$ s com 1024 regras. A *trie* e a híbrida, por outro lado, permanecem praticamente estáveis nesse mesmo

Tabela 2. Cenário de variação no número de regras. `wall_static` e `avg_aspa` em segundos; `rss_mb` em MB.

Div	Regras	Caso	wall_static	rss_mb	avg_aspa
0%	2	original	0.0073 ± 0.0002	20.65 ± 0.07	$(1.392 \pm 0.029)\mathrm{e}{-7}$
0%	2	linear-hit	0.0073 ± 0.0000	20.68 ± 0.08	$(1.464 \pm 0.011)\mathrm{e}{-7}$
0%	2	trie-hit	0.0084 ± 0.0005	20.90 ± 0.05	$(1.880 \pm 0.070)\mathrm{e}{-7}$
0%	2	hybrid-hit	0.0075 ± 0.0001	20.94 ± 0.05	$(1.477 \pm 0.016)\mathrm{e}{-7}$
0%	64	original	0.0076 ± 0.0001	20.63 ± 0.07	$(1.383 \pm 0.013)\mathrm{e}{-7}$
0%	64	linear-hit	0.0120 ± 0.0001	20.78 ± 0.08	$(5.698 \pm 0.050)\mathrm{e}{-7}$
0%	64	trie-hit	0.0083 ± 0.0002	20.89 ± 0.05	$(1.854 \pm 0.013)\mathrm{e}{-7}$
0%	64	hybrid-hit	0.0080 ± 0.0001	20.96 ± 0.06	$(1.757 \pm 0.011)\mathrm{e}{-7}$
0%	1024	original	0.0074 ± 0.0001	20.63 ± 0.07	$(1.378 \pm 0.009)\mathrm{e}{-7}$
0%	1024	linear-hit	0.0765 ± 0.0008	21.49 ± 0.06	$(6.882 \pm 0.038)\mathrm{e}{-6}$
0%	1024	trie-hit	0.0086 ± 0.0003	21.89 ± 0.08	$(2.119 \pm 0.018)\mathrm{e}{-7}$
0%	1024	hybrid-hit	0.0081 ± 0.0001	21.87 ± 0.09	$(1.999 \pm 0.016)\mathrm{e}{-7}$
50%	2	original	0.0111 ± 0.0001	30.86 ± 0.08	$(3.140 \pm 0.044)\mathrm{e}{-7}$
50%	2	linear-hit	0.0130 ± 0.0011	40.21 ± 0.07	$(5.105 \pm 1.067)\mathrm{e}{-7}$
50%	2	trie-hit	0.0570 ± 0.0055	82.86 ± 0.09	$(4.640 \pm 0.487)\mathrm{e}{-6}$
50%	2	hybrid-hit	0.0142 ± 0.0006	42.21 ± 0.05	$(5.902 \pm 0.524)\mathrm{e}{-7}$
50%	64	original	0.0114 ± 0.0002	30.92 ± 0.07	$(3.282 \pm 0.094)\mathrm{e}{-7}$
50%	64	linear-hit	0.0607 ± 0.0025	265.31 ± 0.09	$(4.461 \pm 0.199)\mathrm{e}{-6}$
50%	64	trie-hit	0.1637 ± 0.0033	347.66 ± 0.09	$(1.448 \pm 0.027)\mathrm{e}{-5}$
50%	64	hybrid-hit	0.0589 ± 0.0013	267.35 ± 0.05	$(4.316 \pm 0.101)\mathrm{e}{-6}$
50%	1024	original	0.0109 ± 0.0001	30.89 ± 0.05	$(3.043 \pm 0.047)\mathrm{e}{-7}$
50%	1024	linear-hit	0.5330 ± 0.0056	3756.51 ± 0.10	$(4.088 \pm 0.039)\mathrm{e}{-5}$
50%	1024	trie-hit	1.7692 ± 0.1253	4736.09 ± 0.06	$(1.586 \pm 0.124)\mathrm{e}{-4}$
50%	1024	hybrid-hit	0.5565 ± 0.0132	3758.70 ± 0.10	$(4.276 \pm 0.086)\mathrm{e}{-5}$

intervalo, inclusive no extremo de 1024 regras, com `avg_aspa` de $2{,}1\mathrm{e}{-7}$ s e $2{,}0\mathrm{e}{-7}$ s, respectivamente. Quando a diversidade sobe para 50%, a *trie* volta a perder eficiência, pois precisa ser construída e mantida para um número muito maior de origens. Nesse regime, a híbrida se aproxima da variante linear e se mantém muito abaixo da trie. Em 50% de diversidade e 1024 regras prefixadas por origem, `linear-hit` atinge 0.5330 s de `wall_static`, `hybrid-hit` 0.5565 s e `trie-hit` 1.7692 s. O mesmo padrão aparece em memória: `rss_mb` chega a 4736.09 MB na trie, contra 3756.51 MB na linear e 3758.70 MB na híbrida. Esse resultado reforça a interpretação de que a política híbrida representa um bom compromisso entre os dois extremos.

6.3. Cenários Parametrizados por Métricas do Route Views

Este conjunto, com 1 milhão de rotas, comprimento de caminho igual a 4, 10 provedores autorizados por regra, 2 a 16 regras prefixadas por origem e diversidade de origem de 1% e 2%, aproxima melhor o perfil de carga de interesse do trabalho. Nesse cenário, a discussão deixa de ser sobre extremos artificiais e passa a ser sobre o comportamento das variantes em um regime mais plausível de operação.

Os resultados da Tabela 3 mostram um comportamento mais contido do que aquele observado nos cenários sintéticos extremos. Quando as regras prefixadas entram de fato no caminho de validação, a *trie* apresenta maior custo, enquanto a variante híbrida evita aproximar-se desse extremo e também não acompanha os piores casos observados na

Tabela 3. Cenário parametrizado com 1 milhão de rotas. `wall_static` e `avg_aspa` em segundos; `rss_mb` em MB.

Div	Regras	Caso	wall_static	rss_mb	avg_aspa
1%	2	original	0.9021 ± 0.0322	1823.27 ± 0.08	$(1.529 \pm 0.013)\mathrm{e}{-7}$
1%	2	linear-hit	0.9259 ± 0.0215	1857.36 ± 0.07	$(1.827 \pm 0.019)\mathrm{e}{-7}$
1%	2	trie-hit	1.0564 ± 0.0188	1955.11 ± 0.08	$(2.886 \pm 0.043)\mathrm{e}{-7}$
1%	2	hybrid-hit	0.9391 ± 0.0163	1874.13 ± 0.06	$(1.847 \pm 0.020)\mathrm{e}{-7}$
1%	8	original	0.8601 ± 0.0129	1823.35 ± 0.09	$(1.518 \pm 0.009)\mathrm{e}{-7}$
1%	8	linear-hit	0.9631 ± 0.0219	1900.91 ± 0.07	$(2.218 \pm 0.022)\mathrm{e}{-7}$
1%	8	trie-hit	1.0908 ± 0.0208	1998.74 ± 0.04	$(3.065 \pm 0.065)\mathrm{e}{-7}$
1%	8	hybrid-hit	0.9732 ± 0.0135	1917.63 ± 0.10	$(2.198 \pm 0.007)\mathrm{e}{-7}$
1%	16	original	0.8759 ± 0.0200	1823.32 ± 0.05	$(1.526 \pm 0.016)\mathrm{e}{-7}$
1%	16	linear-hit	1.0349 ± 0.0251	1959.18 ± 0.07	$(2.850 \pm 0.029)\mathrm{e}{-7}$
1%	16	trie-hit	1.1079 ± 0.0131	2056.85 ± 0.08	$(3.200 \pm 0.015)\mathrm{e}{-7}$
1%	16	hybrid-hit	1.0382 ± 0.0130	1975.99 ± 0.05	$(2.778 \pm 0.021)\mathrm{e}{-7}$
2%	2	original	0.9036 ± 0.0204	1842.75 ± 0.06	$(1.624 \pm 0.019)\mathrm{e}{-7}$
2%	2	linear-hit	0.9355 ± 0.0167	1910.58 ± 0.07	$(1.955 \pm 0.008)\mathrm{e}{-7}$
2%	2	trie-hit	1.1463 ± 0.0085	2094.40 ± 0.06	$(3.586 \pm 0.009)\mathrm{e}{-7}$
2%	2	hybrid-hit	0.9639 ± 0.0118	1932.36 ± 0.05	$(2.002 \pm 0.021)\mathrm{e}{-7}$
2%	8	original	0.8848 ± 0.0157	1842.81 ± 0.09	$(1.611 \pm 0.010)\mathrm{e}{-7}$
2%	8	linear-hit	1.0076 ± 0.0211	1997.82 ± 0.09	$(2.393 \pm 0.013)\mathrm{e}{-7}$
2%	8	trie-hit	1.2253 ± 0.0229	2181.66 ± 0.06	$(3.994 \pm 0.063)\mathrm{e}{-7}$
2%	8	hybrid-hit	1.0281 ± 0.0160	2019.58 ± 0.06	$(2.412 \pm 0.012)\mathrm{e}{-7}$
2%	16	original	0.8917 ± 0.0309	1842.78 ± 0.11	$(1.632 \pm 0.039)\mathrm{e}{-7}$
2%	16	linear-hit	1.0938 ± 0.0162	2114.17 ± 0.08	$(3.060 \pm 0.007)\mathrm{e}{-7}$
2%	16	trie-hit	1.3085 ± 0.0518	2297.96 ± 0.07	$(4.506 \pm 0.217)\mathrm{e}{-7}$
2%	16	hybrid-hit	1.0884 ± 0.0120	2136.02 ± 0.07	$(3.009 \pm 0.009)\mathrm{e}{-7}$

busca linear quando o número de regras por origem cresce. Em 1% de diversidade de origem e 16 regras prefixadas por origem, por exemplo, `linear-hit` registra 1.0349 s de `wall_static`, enquanto `hybrid-hit` registra 1.0382 s. Nesse mesmo ponto, os valores de `avg_aspa` também permanecem próximos, com $2{,}78\mathrm{e}{-7}$ s na híbrida e $2{,}85\mathrm{e}{-7}$ s na linear. Em 2% de diversidade de origem e 16 regras por origem, a mesma tendência se mantém: `hybrid-hit` registra 1.0884 s de `wall_static`, contra 1.0938 s da linear, e $3{,}01\mathrm{e}{-7}$ s de `avg_aspa`, contra $3{,}06\mathrm{e}{-7}$ s. A trie, por sua vez, permanece claramente mais cara ao longo de todo o conjunto parametrizado, tanto em tempo quanto em memória.

7. Trabalhos Relacionados

Várias propostas recentes buscam solucionar o problema da validação de rotas em anúncios BGP. BGPSec é uma extensão de BGP na qual cada AS assina criptograficamente o anúncio repassado ao próximo vizinho para impedir modificações no AS-path [Lepinski and Sriram 2017]. Apesar de suas fortes propriedades teóricas, BGPsec enfrenta barreiras significativas à implantação devido ao custo computacional elevado e aos benefícios limitados em caso de adoção parcial [Furuness et al. 2025].

O ASPA é uma proposta alternativa mais leve que BGPSec para validar rotas com base em informações de relacionamento provedor-cliente publicadas em RPKI

[Azimov et al. 2024, Azimov et al. 2025]. Sua ideia central é permitir que um AS autorize criptograficamente seus provedores, de modo que o caminho anunciado possa ser verificado por um modelo compatível com a propagação esperada de rotas. Trabalhos recentes mostram que o ASPA é especialmente relevante em um cenário de maior adoção de ROV, no qual ataques pós-ROV e vazamentos de rota se tornam mais relevantes [Furuness et al. 2025]. Furuness *et al.* [2025] discutem também o ASPAwN, uma extensão proposta para aumentar a cobertura do ASPA.

Outra direção recente é o ASRA (*Autonomous System Relationship Authorization*), proposta voltada a complementar o ASPA por meio do registro explícito de relações adicionais (*e.g.,* clientes e pares) com o objetivo de reduzir lacunas conhecidas da validação baseada apenas em provedores [NIST 2026]. O PAVA (*PAth VAlidation*) propõe validar o AS-path consultando os ASes ao longo do caminho sobre seus relacionamentos, utilizando uma infraestrutura de consultas apoiada em DNS/DNSSEC [Flechier et al. 2026]. Apesar de também buscar maior expressividade, o PAVA segue uma direção distinta da adotada por PPASPA, pois não estende o modelo atual da RPKI e depende de uma infraestrutura de consultas distribuídas. Já a proposta baseada em RPA (*Route Path Authorization*) busca representar e verificar descrições autorizadas de propagação de rotas por meio de novos objetos RPKI [Guo et al. 2025, Xu et al. 2025]. Entre as propostas recentes, o RPA é conceitualmente mais próxima do PPASPA, pois também busca aumentar a expressividade do modelo de autorização dentro do ecossistema RPKI. Ainda assim, a diferença central é que PPASPA preserva a lógica e a estrutura básica de ASPA e refina apenas a seleção da autorização aplicável ao primeiro enlace do caminho, enquanto RPA propõe novos objetos e um modelo de verificação mais amplo para descrever caminhos autorizados. Essas abordagens diferem do ASPA tanto no tipo de informação publicada quanto no modelo de verificação empregado.

Diferentemente dessas propostas, este trabalho não introduz um novo objeto criptográfico nem um novo algoritmo de validação de caminho. Sua contribuição está na proposta, implementação e avaliação experimental de PPASPA que permite a especificação de prefixos em objetos ASPA. Portanto, espera-se que PPASPA seja mais fácil de ser adotado por partir de uma proposta já amplamente discutida na comunidade.

8. Conclusão

Este trabalho apresentou PPASPA, uma extensão de ASPA que permite granularidade por prefixo na autorização de provedores, possibilitando que um sistema autônomo associe diferentes conjuntos de provedores autorizados a prefixos específicos. A proposta preserva a lógica original de validação de ASPA e modifica apenas o mecanismo de seleção da regra aplicável ao enlace entre a origem e seu provedor direto, mantendo compatibilidade com o comportamento tradicional por meio de uma regra global de *fallback*.

Implementamos PPASPA no *software* de roteamento BIRD 2 reutilizando estruturas de dados já presentes no sistema, em particular listas lineares e árvores de prefixo (*tries*). Os resultados experimentais variando o número de origens distintas e a quantidade de regras prefixadas por origem mostram que a extensão é viável e introduz baixo custo computacional. Os experimentos mostraram que a abordagem híbrida baseada em busca linear e *tries* é promissora para acomodar diferentes regimes de carga.

Agradecimentos

O presente trabalho foi realizado com o apoio do Programa IETF do Comitê Gestor da Internet (CGI.br), da CAPES – Código de Financiamento 001, do CNPq – Procs. 312371/2026-8, 407302/2025-5, 420934/2023-5, 308101/2022-7, da FAPESP – Procs. 2023/00812-7 e 2023/00811-0, e FAPEMIG APQ-02793-23. Utilizamos a ferramenta de IA ChatGPT para correções gramaticais e aperfeiçoamento do texto.

Referências

Azimov, A., Bogomazov, E., Bush, R., Patel, K., Snijders, J., and Sriram, K. (2025). BGP AS PATH Verification Based on Autonomous System Provider Authorization (ASPA) Objects. Internet-Draft draft-ietf-sidrops-aspa-verification-22, IETF. Work in Progress.

Azimov, A., Uskov, E., Bush, R., Snijders, J., Housley, R., and Maddison, B. (2024). A Profile for Autonomous System Provider Authorization. Internet-Draft draft-ietf-sidrops-aspa-profile-18, IETF. Work in Progress.

Blunk, L., Damas, J., O'Donoghue, F., Gittings, B., Johns, M. S., and Sobrinho, J. L. (2005). Routing Policy Specification Language next generation (RPSLng). RFC 4012, IETF.

Borchert, O., Lee, K., Sriram, K., Montgomery, D., Gleichmann, P., and Adalier, M. (2021). Bgp secure routing extension (bgp-srx): Reference implementation and test tools for emerging bgp security standards. Technical Report NIST TN 2060, National Institute of Standards and Technology.

Bush, R. and Austein, R. (2017). The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 1. RFC 8210, IETF.

Butler, K., Farley, T., McDaniel, P., and Rexford, J. (2004). A survey of bgp security. *ACM, draft version*, 5:1–35.

Butler, K., Farley, T. R., McDaniel, P., and Rexford, J. (2010). A survey of BGP security issues and solutions. *Proceedings of the IEEE*, 98(1):100–122.

CZ.NIC (2026). Bird internet routing daemon. Acessado em: 26 de abr. de 2026.

Flechier, M., Heusse, M., and Duda, A. (2026). PAVA: BGP AS_PATH Validation by Querying ASes about Their Relationships. Internet-Draft draft-flechier-sidrops-pava-01, IETF. Work in Progress.

Furuness, J., Morris, C., Morillo, R., Kasiliya, A., Wang, B., and Herzberg, A. (2025). Securing BGP ASAP: ASPA and other Post-ROV Defenses. In *Network and Distributed System Security (NDSS) Symposium 2025*. Internet Society.

Gao, L. and Rexford, J. (2001). Stable internet routing without global coordination. *IEEE/ACM Transactions on Networking*, 9(6):681–692.

Giotsas, V., Luckie, M., Huffaker, B., and Claffy, K. (2014). Inferring complex as relationships. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 23–29, New York, NY, USA. ACM.

Guo, Y., Wang, X., Xu, K., Liu, Z., and Li, Q. (2025). A Profile for Route Path Authorizations (RPAs). Internet-Draft draft-guo-sidrops-rpa-profile-00, IETF. Work in Progress.

Jin, Y., Scott, C., Dhamdhere, A., Giotsas, V., Krishnamurthy, A., and Shenker, S. (2019). Stable and practical as relationship inference with problink. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 581–598. USENIX Association.

Kastanakis, S., Giotsas, V., Livadariu, I., and Suri, N. (2023). Replication: 20 years of inferring interdomain routing policies. In *Proceedings of the 2023 ACM on Internet Measurement Conference*, IMC '23, pages 16–29, New York, NY, USA. Association for Computing Machinery.

Khadka, S. K. (2025). A first look at the adoption of bgp-based ddos scrubbing services. https://labs.ripe.net/author/shyam-krishna-khadka/a-first-look-at-the-adoption-of-bgp-based-ddos-scrubbing-services/. RIPE Labs, accessed from the article page.

Khadka, S. K., Bayhan, S., Holz, R., and Hesselman, C. (2026). Detecting and characterizing ddos scrubbing from global bgp routing: Insights from five leading scrubbers. In *Passive and Active Measurement*, volume 16477 of *Lecture Notes in Computer Science*, pages 17–43. Springer, Cham.

Lepinski, M. and Sriram, K. (2017). BGPsec Protocol Specification. RFC 8205.

NIST (2026). Robust Inter-Domain Routing. Acessado em 10 maio 2026.

Rekhter, Y., Li, T., and Hares, S. (2006). A Border Gateway Protocol 4 (BGP-4). RFC 4271, IETF.

Rekhter, Y., Li, T., Karrenberg, D., Terpstra, J., and Yu, J. (1999). Routing Policy Specification Language (RPSL). RFC 2622, IETF.

Rodday, N., Cunha, I., Bush, R., Katz-Bassett, E., Rodosek, G. D., Schmidt, T. C., and Wählisch, M. (2024). The resource public key infrastructure (RPKI): A survey on measurements and future prospects. *IEEE Transactions on Network and Service Management*, 21(2):2353–2373.

Testart, C., Wolff, J., Gouda, D., and Fontugne, R. (2024). Identifying current barriers in RPKI adoption. Technical report, Georgia Institute of Technology and Tufts University and IIJ Research Laboratory. Pre-print manuscript.

University of Oregon (2026). University of Oregon Route Views Project. Acessado em: 27 de janeiro de 2026.

Xu, K., Jiang, S., Guo, Y., and Wang, X. (2025). BGP AS_PATH Verification Based on Route Path Authorizations (RPA) Objects. Internet-Draft draft-xu-sidrops-rpa-verification-01, IETF. Work in Progress.