Computer Security Symposium 2020
26 - 29 October 2020

# APVAS+: 集約署名を用いたAS パス検証プロトコルの改良とフルルート情報を想定した実験評価

竹村 達也[1]　矢内 直人[1]　梅田 直希[1]　岡田 雅之[3]　岡村 真吾[2]

**概要**：BGPsec は電子署名を利用してインターネット経路情報の正当性を保証するプロトコルであるが、電子署名によるメモリ量の負荷が問題だった。近年に複数の署名を一定サイズの単一の署名に集約する集約署名を用いることで BGPsec のメモリ量を削減するプロトコルが提案されたが、 このプロトコルも依然として削減量が不十分である。本稿ではメモリ量を現実的に運用可能なサイズまで削減したプロトコル *APVAS+* を提案する。大まかには既存のプロトコルがひとつの線状ネットワークの署名しか集約できないことに対し、APVAS+ では複雑なネットワークトポロジの署名も集約可能となっている。また、この実現に向けて新たな集約署名を設計した。また、ソフトウェアルータ BIRD を拡張して APVAS+ をプロトタイプ実装した。さらに、世界規模の経路（フルルート）情報に相当する約 80 万経路を想定した実験評価を行い、従来の BGPsec と比較してメモリ長が最大 8 割削減されることを確認している。

**キーワード**：BGPsec, APVS、集約署名、BIRD、フルルート実験

# A Study on AS_PATH Validation Protocol Based on Aggregate Signatures and Its Experimental Evaluations

Tatsuya Takemura[1]　Naoto Yanai[1]　Naoki Umeda[1]　Masayuki Okada[3]　Shingo Okamura[2]

***Abstract:*** BGPsec is a protocol to guarantee the validity of routing information on the Internet by utilizing digital signatures whereas it is impractical due to the heavy size of a memory storage. Although an extension of BGPsec based on an aggregate signature scheme, which enable to aggregate individual signatures into a single short signature, was proposed in recent years, it was still impractical. In this paper, we design a new protocol named *APVAS+*, whose memory size can be reduced to an operable size in the real world. Loosely speaking, while the existing protocol allows to aggregate signatures on only linear networks, APVAS+ can aggregate those even on more complicated networks. To do this, we also present a novel aggregate signature scheme. We show a prototype implementation of APVAS+ by extending a router software called BIRD as well. Moreover, we conduct experiments on the full route information, i.e., about 800,000 routes, and then confirm that the memory size can be reduced by 80 percentages in comparison with the existing protocols.

***Keywords:*** BGPsec, APVAS, Aggregate Signatures, BIRD, Full Route Experiment

## 1. はじめに

**背景）** *Border Gateway Protocol security extension (BGPsec)* [14] は、自律システム (AS) という単位でインターネット経路を制御するプロトコルである *Border Gateway Protocol (BGP)* [20] に対し電子署名を導入することで、インターネット経路の正当性を保証するプロトコルである。従来の BGP は、経路情報の正当性を保証できないことから、誤った AS の情報や AS 間の接続情報を受信した際に、経路を乗っ取られる危険性が知られていた[*1]。

[1] 大阪大学. Osaka University
[2] 奈良工業高等専門学校. National Institute of Technology, Nara College
[3] 長崎県立大学. University of Nagasaki

[*1] https://www.ripe.net/publications/news/industry-developments/

このような事件は被害規模が大きい一方、実態調査 [26] によれば一日平均 4 件と発生頻度が高い。このような歴史的背景から、電子署名を導入することで経路の正しさを保証する BGPsec の登場はもはや必然といえる [12].

その一方で、BGPsec は主な問題として電子署名の導入によりルータメモリが枯渇する**メモリ枯渇問題**が指摘されている [23]. 直観的には電子署名により各 AS の情報が百倍以上増幅されることに加え、各経路毎およびその経路上に存在する AS 毎に電子署名が生成されることから、数十 GB ものメモリが必要となる。これは現状のルータ性能を大幅に逸脱しており、未だに実用化のめどがたっていない。

近年、このメモリ枯渇問題に対し、独立に生成された複数の電子署名を等価な単一の署名に集約する集約署名 [4] を BGPsec に導入したプロトコル *APVAS* [11], [24] が提案された。直観的には、署名の集約を通じてメモリ負荷を AS 台数から独立させることで、メモリ枯渇問題の解決を図っている。しかしながら、APVAS は互いに異なる経路同士の電子署名を集約する機能は持たない。すなわち、APVAS は単一の経路上で署名を集約する機能しか持たない。このため、メモリ量を高々 30% しか削減できておらず、未だルータ性能が大幅に不足したままである。すなわち、依然としてメモリ枯渇問題を解決するには至っていない。

**貢献）**本稿では互いに独立した経路の署名も集約できるプロトコル *APVAS+* を提案する。また、実際の世界規模のインターネット経路（フルルート）情報を用いた実装実験を行うことで、社会実装された際の性能も検討する。

本稿の技術的貢献は三点である。まず互いに異なる経路情報を集約・広告する機能を設計したことである。集約署名により理論上署名の集約自体は可能である一方、BGPsec（および BGP）は要求された宛先に対する最適経路 (best path) のみ経路広告する機能しか持たない。すなわち、署名を集約することでその最適経路に対する元の署名が失われるため、BGPsec 本来の経路広告機能が損なわれてしまう。これは APVAS [11] においても同様であり、**単純な拡張では APVAS+ の機能が実現できない**ことを意味する。本稿ではこの問題を回避する要素技術として**楽観的併存型集約署名 (Optimistic Bimodal Aggregate Signature)** を新たに設計・導入することで、経路の集約と広告を両立する機能を実現した。（詳細は 4 に記載する。）

第二の貢献として、APVAS+をソフトウェアルータ BIRD[*2] を拡張することでプロトタイプ実装した。これにより APVAS+ を 実機で動作させ、その現実世界における性能を確認できるようになった。（詳細は 7 節に記載する。）

第三に、フルルート情報に相当する経路情報を用いた実験により、APVAS+ が世界規模で利用された際の性能を実機で初めて示した。フルルート情報自体は公開されているが、集約署名など高機能な暗号技術を用いたプロトコルでは、フルルート情報を想定した実機による実験は、認識している限り初めてである。これは大雑把な評価ではあるが、結果として、実際に**実用的な性能に収まることを世界で初めて実証**した。（詳細は 8 節に記載する。）

youtube-hijacking-a-ripe-ncc-ris-case-study, https://www.internetsociety.org/blog/2018/04/amazons-route-53-bgp-hijack/

[*2] The BIRD Internet Routing Daemon: https://bird.network.cz/

## 2. 関連研究

本節では関連研究として BGPsec の拡張研究および BGP セキュリティの研究について紹介する。また、集約署名など高機能暗号の BGP 応用についても述べる。

**BGPsec の応用研究：**本稿に最も近い研究は集約署名 [4] を用いた BGPsec の拡張プロトコルの設計 [11], [24], [28] である。このうち実機への実装およびメモリ評価を行った成果は APVAS [11] のみであるが、前述した通り、APVAS も線状ネットワークでの評価しか行えていない。また、性能も高々三割程度のメモリ負荷を削減したのみである。なお、BGPsec の導入実装としては、専用の公開鍵管理基盤 (PKI) であるリソース PKI (RPKI) [13] が標準化されており、既に普及が進んでいる [6]。一方、RPKI 及び BGPsec は部分的な導入では未だ攻撃ができることも示されており [16], [18]、完全な導入が望まれる。

**BGP セキュリティの現状：**近年では BGP への更なる攻撃の調査 [3], [9], [18] も発展が著しく、BGP の攻撃を介して暗号通貨を盗む攻撃 [7], [22], [25] も確認された。また、近年では DoS 攻撃などサイバー攻撃を緩和させるために、あえて BGP の経路ハイジャックを行うブラックホールサービスの普及も進んでいる [8], [19]。これらの攻撃含め実世界の BGP の運用は、BGPsec の導入が実現すれば制御が可能である [21]。

**高機能暗号の BGP 応用：**集約署名の理論研究 [4], [5], [10], [17] では BGP への応用が言及されているが、実装実験は行われていない。また、BGP に入力を秘匿したまま計算を行う秘密計算を導入しポリシーを隠す成果 [2] もあるが、やはり実機による実装実験は行われていない。

## 3. 問題設定

本節では、BGPsec の機能および APVAS [11] について述べる。また、本稿の主な問題設定として、APVAS 含めた既存技術の限界について述べる。

### 3.1 BGPsec

BGPsec [14] は，インターネット全体での宛先問題をルータおよび IP アドレスの集合である自律システム (AS) 単位で解決するプロトコルである BGP [20] に電子署名を導入することで、経路情報の正当性を確認できるプロトコルである。各 AS はそれぞれ一意に割り当てられた AS 番号

で各ネットワークを区別し、その AS が保有する IP プレフィックスと各宛先への経路を他の AS と交換することで、AS 間で経路を構成する。この交換処理を経路広告と呼び、経路広告されてきた経路をルータ内の経路表に格納する。

経路広告は、利用可能な経路情報を定義する Network Layer Reachability Information (NLRI) とその宛先に向けた BGPsec_PATH を含む update メッセージの送受信によって実現される [14]。ここで BGPsec_PATH 属性は、Secure_PATH と Signature_Block によって構成される。Secure_PATH は経路情報が通過してきた各 AS の AS 番号をリスト化したものであり、Signature_Block は Secure_PATH の中の各 AS が付加した電子署名を格納するところである。より正確には、Signature_Block は Signature_Block Length、Sequence of Signature Segments を含んでおり、このうち Sequence of Signature Segments が AS 台数に依存する可変長になる。また、Sequence of Signature Segments の中には公開鍵識別子 (SKI) を表す Subject Key Identifier と、署名に関連する情報として Signature Length、Signature が格納される。一方、経路が失効される withdrawn 処理も同様に update メッセージが利用され、失効する経路情報を保証する BGPsec_PATH 属性が含まれる。以降では単に経路情報と書いた場合、上述した経路広告される情報すべてを含むものとする。なお、BGPsec は Secure_PATH 内のリストの保護が主な機能であり、IP プレフィックス自体は RPKI [13] により保証される。

BGPsec のメモリ枯渇問題とは、他のルータによって経路広告される経路情報により、ルータ内に保存される経路表のデータサイズが膨大となることを意味する。

### 3.2 APVAS

APVAS [11] は集約署名 [4] を用いることで、BGPsec における Signature_Block 内の署名の数が固定になるようにしている。より正確には併存型集約署名と呼ばれる、付録 5 に記載する方式を用いている。このとき、APVAS の Signature_Block は Signature_Block Length, Algorithm Suite Identifier, 集約署名に関連する Signature Length と Signature, および各 AS の SKI を格納する Subject Key Identifier Segments で構成される。このうち、Signature_Block Length と Algorithm Suite Identifier は BGPsec と同様に定義される。直観的には、従来の BGPsec は Sequence of Signature Segments 自体が AS に依存して増えることに対し、APVAS では集約署名により署名が固定情報になることで Signature Length と Signature が固定長になり、SKI のみが AS に依存する可変長のセグメント Subject Key Identifier Segments として定義される。

### 3.3 APVAS 含めた既存の技術的問題

APVAS の欠点は線状のトポロジしか対応できないことである。つまり、異なる BGPsec_PATH 間の署名は集約できず、Secure_PATH の数に電子署名の数が依存する。例えば、80 万本の異なる Secure_PATH があると 80 万個の署名が必要になることもあり得る。これは依然として実用的な性能にならず、やはりメモリ枯渇問題を解決できていない。

上述した APVAS の問題は、集約署名と BGPsec のそれぞれの仕様に関して、未だ満足な互換性が得られていないことに起因する。まず、集約署名は理論上は全ての署名を集約できるため、直観としては異なる経路同士の署名も集約可能できるようにみえる。しかし、異なる Secure_PATH の署名を集約することに起因して、経路広告機能が損なわれてしまう。具体的には、以下の二点の問題が生じる。

まず、BGPsec（および BGP）は問い合わせされた宛先に対し、最適経路のみ保存・経路広告する最適経路選択を採用している。このとき、異なる Secure_PATH の署名を集約することによって、最適経路に相当する署名が失われてしまう。言い方を変えると、集約署名は最適経路に対する元の署名は含んでいるが、その署名だけを取り出すような機能は持たない。仮に集約署名を用いて最適経路だけに関する検証を試みても、検証は不受理となる。

第二に、一度集約された署名は分割ができないことから、署名を集約済みの Secure_PATH に対しては経路を失効する withdrawn 処理が行えなくなる。BGPsec（および BGP）では一部の経路が故障などで利用できなくなった場合や特定の経路が管理対象ではなくなった場合に該当する Secure_PATH を経路表から除外する。一度 Secure_PATH が失効されると、該当する署名とともに集約された他の署名の検証も行うことができなくなる。

なお、集約署名の一般的問題として、検証が不受理になる署名が集約されることで他の署名も不受理になる署名巻き添え問題 [10], [24] があるが、これはそれぞれの論文で示されたトレーシングにより解決できる。このため、本稿では扱わない。

## 4. APVAS+ の設計方針

本節では前節で述べた問題を解決するプロトコル APVAS+ の提案にあたり、その設計方針を述べる。APVAS+ の主な着想は「集約署名の検証が成立するならば、含まれている経路の正当性はすべて確認ができる」という点にある。この観点から、APVAS+ の要素技術として次節に示す楽観的併存型集約署名を新たに設計した。大まかには、楽観的併存型集約署名は集約された署名の検証式の一部をさらに署名に変換することで、集約された状態から一部の署名だけ検証できる機能を提供する。これにより、集約署名だけがある状態で、最適経路に相当する Secure_PATH と Signature_Block の情報が検証できる。また、withdrawn

処理の実現については、楽観的併存型集約署名のアルゴリズムを拡張することで、集約された経路から withdrawn 対象となる経路情報を削除することで実現する。

詳細は 6 節に記載するが、次節に示す楽観的併存型集約署名のもとで APVAS+ のプロトコル仕様を設けた。

## 5. 楽観的併存型集約署名

本節では APVAS+ の要素技術として楽観的併存型集約署名 (Optimistic Bimodal Aggregate Signature, OBAS) を提案する。本方式は Secure_PATH 内の署名の集約と Secure_PATH 間の署名の集約に加え、withdrawn 処理に際し失効された Secure_PATH の署名を除去する機能及び最適経路の経路広告時に該当する Secure_PATH の署名のみを集約署名から検証する機能を提供する。本節では Secure_PATH と署名をあわせて署名チェーンと呼ぶ。

楽観的併存型集約署名は次に定義する双線形写像を用いる。$\mathbb{G}$, $\mathbb{G}_T$ を素数位数 $p$ を持つ群とする。このとき、双線形写像 $\boldsymbol{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ は以下の条件を持つ写像とする。まず任意の $U, V \in \mathbb{G}$ および $a, b \in \mathbb{Z}_p^*$ において、$\boldsymbol{e}(aU, bV) = \boldsymbol{e}(U, V)^{ab}$ となる。次に、任意の生成元 $P \in \mathbb{G}$ において、$\boldsymbol{e}(P, P) \neq 1_{\mathbb{G}_T}$ となる。ここで，$1_{\mathbb{G}_T}$ は $\mathbb{G}_T$ 上の単位元とする。最後に、任意の $U, V \in \mathbb{G}$ において、効率的に $\boldsymbol{e}(U, V)$ を計算可能である。また、本稿では群 $\mathbb{G}$, $\mathbb{G}_T$ 上の離散対数問題 (DLP) は困難とする。このとき、上述の条件を満たす $\mathbb{G}$ を双線形群と呼び、そのパラメータ $(p, \mathbb{G}, \mathbb{G}_T, \boldsymbol{e})$ を双線形パラメータと呼ぶ。

また。以下では各署名者は互いに一意な識別子番号 $i$ を持つものとする。任意の署名において関与している署名者の公開鍵と平文のリストを $L = \{(pk_i, m_i)\}$ とし、その識別子の集合を $S$ とする。任意の $i \in S$ に関して $i$ 番目の署名者までの署名チェーンに参加する署名者の集合を $S_i \subseteq S$ とする. 文字列間の連結を $\|$ とし。特に $S_i$ に属する署名者が持つ文字列間の連結を任意の $j$ を用いて $\|_{j \in S_i}$ で表す。

**Algorithm 1-7** に方式を記載する。**Algorithm 3** の 3 行目が Secure_PATH 内の署名の集約、**Algorithm 4** の 1 行目が Secure_PATH 間の署名の集約にそれぞれ相当する。これらの処理は併存型集約署名 [11] に着想を得た。

楽観的併存型集約署名はさらに集約署名の検証式を署名の一部に変換する機能と、それにより集約署名の一部だけを検証する機能を持つ。前者の機能は **Algorithm 6** の 4 行目から 7 行目、後者の機能は **Algorithm 7** の 5 行目から 9 行目にそれぞれ相当する。ここで生成・利用される $E$ は **Algorithm 5** の 3 行目にも相当し、検証時に署名チェーンを反映する部分となっている。また、**Algorithm 6** の出力により署名の要素として $E$ が追加されるが、$E$ のサイズは群 $\mathbb{G}_T$ の要素一つ分である。すなわち、署名サイズを定数オーダに抑える集約署名の性質には矛盾しない。

また、**Algorithm 1-7** に記載した方式の安全性は，形式的に定義と証明を行うことが可能である．紙面の都合上，詳細は省略するが，ランダムオラクルモデルにおいて CDH 仮定の下で偽造不可能性を証明できる．

**Algorithm 1** Setup

**Require:** セキュリティパラメータ $1^\kappa, 1^\tau$
**Ensure:** グローバルパラメータ $para$
1: 双線形パラメータ $(p, \mathbb{G}, \mathbb{G}_T, \mathbf{e})$ を生成
2: $P \leftarrow \mathbb{G}$
3: ハッシュ関数 $H : \{0,1\}^* \to \mathbb{G}$ を選択
4: $para = (p, \mathbb{G}, \mathbb{G}_T, \mathbf{e}, P, H)$

**Algorithm 2** UserKeyGen

**Require:** グローバルパラメータ $para$
**Ensure:** 秘密鍵 $sk$，公開鍵 $pk$
1: 乱数 $sk \leftarrow \mathbb{Z}_p$ を生成
2: $X = xP$
3: $sk = x$，$pk = X$

**Algorithm 3** SeqAggSign

**Require:** グローバルパラメータ $para$，秘密鍵 $sk_i$，公開鍵 $pk_i$，平文 $m_i \in \{0,1\}^*$，公開鍵-平文リスト $L = \{(pk_j, m_j)\}_{j \in S}$，署名 $\sigma$
**Ensure:** 署名 $\sigma$，公開鍵-平文リスト $L' = \{(pk_j, m_j)\}_{j \in S} \cup \{(pk_i, m_i)\}$
1: $sk_i$ を $x_i$ として構文解析
2: $L = \emptyset$ の場合，$\sigma = 0$ と設定
3: $c = H\left(\mathbf{e}(\sigma, P) \parallel pk_i \parallel m_i \parallel_{j \in S_i} (pk_j \parallel m_j)\right)$
4: $\sigma = \sigma + x_i \cdot c$

**Algorithm 4** AggSign

**Require:** グローバルパラメータ $para$, 公開鍵-平文リストと署名の組 $\{(L_i, \sigma_i)\}_{i \in [1,n]}$
**Ensure:** 署名 $\sigma$, 公開鍵-平文リスト $L' = L_1 \cup \cdots, \cup L_n$
1: $\sigma = \sum_{i \in [1,n]} \sigma_i$

さらに、上述した楽観的併存型集約署名は、その機能を拡張することで、 集約署名から失効された署名を取り除く機能として **RemSign** を導入できる。**RemSign** は divide-out 技法 [15] に基づいて方式を拡張することで具体化される。詳細については紙面上割愛するが、大まかには **Algorithm 3** の処理が決定的アルゴリズムであることから、該当する署名チェーンを再計算すること、また、**Algorithm 4** で署名同士の足し算の代わりに引き算をすることで、withdrawn 処理が可能となる。

## 6. APVAS+ のプロトコル仕様

APVAS+ の仕様詳細および動作概要について、図 2 を用いながら、以下に述べる。まず、各 AS は初期設定として Algorithm 1 によってグローバルパラメータの生成と共有が行われる。また、Algorithm 2 によって秘密鍵の生成と対応する公開鍵の他 AS への配布が行われている。ここでは AS1 が 10.0.0.1/24，10.0.0.2/24，10.0.0.3/24 の経路生成元、 AS2 が 10.0.0.4/24，10.0.0.5/24 の経路

**Algorithm 5** Verify

**Require:** グローバルパラメータ $para$, 公開鍵-平文リスト $L = \{(pk_j, m_j)\}_{j \in S}$, 署名 $\sigma$
**Ensure:** True or False
1: **for all** $i \in S$ **do**
2: $pk_i$ を $X_i$ として構文解析
3: $c_i = H\left(\left(\prod_{j \in S_i} \mathbf{e}(c_j, X_j)\right) \parallel pk_i \parallel m_i \parallel_{j \in S_i} (pk_j \parallel m_j)\right)$
4: **end for**
5: **if** $\mathbf{e}(\sigma, P) = \prod_{i \in S} \mathbf{e}(c_i, X_i)$ **then**
6: **return** True
7: **else**
8: **return** False
9: **end if**

**Algorithm 6** PointGen

**Require:** グローバルパラメータ $para$, 公開鍵-平文リスト $L = \{(pk_j, m_j)\}_{j \in S}$, 署名 $\sigma$, 検証対象の識別子リスト $\mathcal{I} \subseteq S$
**Ensure:** 部分検証用署名 $E$
1: **for all** $i \in S$ **do**
2: $pk_i$ を $X_i$ として構文解析
3: **end for**
4: **for all** $i \in S \setminus \mathcal{I}$ **do**
5: $c_i = H\left(\left(\prod_{j \in S_i} \mathbf{e}(c_j, X_j)\right) \parallel pk_i \parallel m_i \parallel_{j \in S_i} (pk_j \parallel m_j)\right)$
6: **end for**
7: $E = \prod_{i \in S \setminus \mathcal{I}} e(c_i, X_i)$

**Algorithm 7** PointVer

**Require:** グローバルパラメータ $para$, 公開鍵-平文リスト $L_i = \{(pk_j, m_j)\}_{j \in S}$, 署名 $\sigma$, 部分検証用署名 $E$, 検証対象者の識別子リスト $\mathcal{I}$
**Ensure:** True or False
1: **for all** $i \in \mathcal{I}$ **do**
2: $pk_i$ を $X_i$ として構文解析
3: $c_i = H\left(\left(\prod_{j \in S_i} \mathbf{e}(c_j, X_j)\right) \parallel pk_i \parallel m_i \parallel_{j \in S_i} (pk_j \parallel m_j)\right)$
4: **end for**
5: **if** $e(\sigma, P) = \prod_{i \in \mathcal{I}} e(c_i, X_i) \cdot E$ **then**
6: **return** True
7: **else**
8: **return** False
9: **end if**

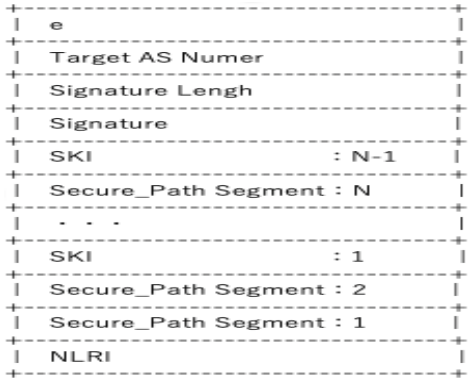


図 1 APVAS+におけるハッシュ対象

生成元である。AS7 などの各中継 AS は受信した update メッセージから経路情報を抽出することで図 1 に示す情報を得る。これらのハッシュ値を用いて、Algorithm 5 によって検証を行う。

次に署名の生成と集約を行うが、ここでの集約は二種類存在する。一つ目はある一経路に対する Secure_PATH 内の署名の集約であり、update メッセージを受け取った AS が Secure_PATH への自身の AS 番号の付加と Algorithm 3 を利用した署名チェーンの生成と集約を行う。二つ目は 1 つの AS が自身の所持する異なる経路情報に対して行う Secure_PATH 間の署名の集約であり、Algorithm 4 を利用する。AS7 では Secure_PATH（AS1, AS2）から受信した 3 本の経路に関する各署名を $\sigma_1$ に、Secure_PATH（AS3, AS4, AS5）から受信した 2 本の経路に関する各署名を $\sigma_2$ に集約したのち、さらに集約して $\sigma$ としている。各経路に関する署名は $\sigma$ を AS8 に送信後、破棄する。これにより、署名のメモリ量を削減している。

以下では APVAS+ で新たに導入する機能として、一部の経路のみ送信する**部分経路広告**と経路を削除する **withdrawn 処理**について、図 2 を用いながら説明する。

まず部分経路広告について述べる。これは署名を集約後に一部のみの経路情報を送信する際の処理であり、楽観的併存型集約署名の部分検証を利用する。AS7 が 10.0.0.2/24 以外の経路を AS8 に送信するとき、AS7 は Algorithm 6 によって部分検証用署名 E を作成し、update メッセージを送信する。update メッセージを受信した AS8 は Algorithm 7 により経路広告された経路のみ検証できる。また、新規参入 AS に対して自身の持つ経路情報を送信する refresh 要求の際も、上述した部分経路広告を用いる。

次に、withdrawn 処理について述べる。AS8 が AS6 から過去に受け取っていた経路 10.0.0.6/24 が利用不可能となったとき、AS8 は AS6 から withdrawn 付きの update メッセージを受け取る。AS8 は、前節で述べた **RemSign**[*3] によって該当する Secure_PATH の署名を消去する。これにより、経路表からも該当する経路情報を消去する。

## 7. 実装

ソフトウェアルータ BIRD とペアリングライブラリ TEPLA[*4] を利用して APVAS+のプロトタイプ実装を行った。BIRD は C 言語で開発されたルーティング・ソフトウェアであり、一台の計算機を一台の BGP ルータとして使用することができる。BGP に関連するソースコードが存在する proto/bgp/以下の packets.c, bgp.h を編集して、update メッセージの仕様変更と楽観的集約署名の各機能を実現した。図 1 の情報を構成してハッシュ値を計算したり署名を生成・集約する機能と、APVAS+仕様の update メッセージにエンコードを作成して送信する機能を bgp_encode_prefixes() 関数に組み込む。また、update

[*3] 前節に記載したとおり、紙面の都合からアルゴリズム自体の掲載は割愛している。
[*4] http://www.cipher.risk.tsukuba.ac.jp/tepla/

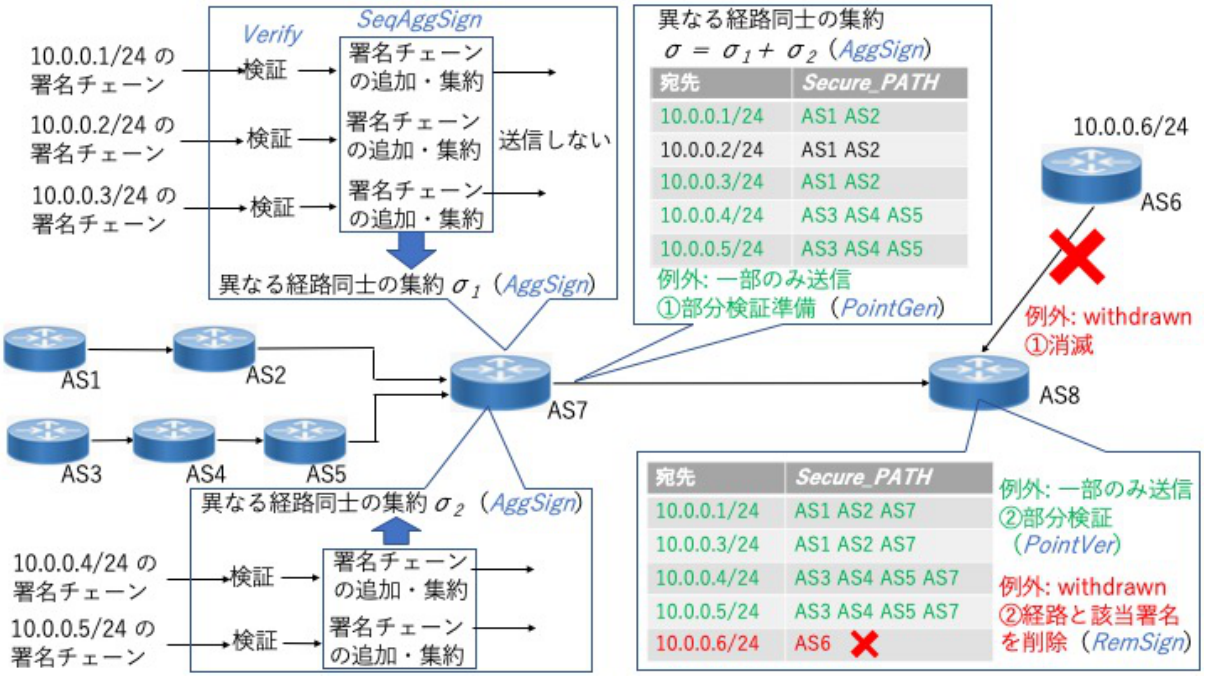


**図 2** APVAS+ の動作の流れ

メッセージを受信してデコードした後、署名の検証を行う機能を `bgp_decode_attrs()` 関数に組み込む。楽観的集約署名の実装において、TEPLA は ver.2.0 を利用する。なお、ペアリング演算において楕円曲線は ECBN254a、有限体は bn254_fpa をパラメータとして用いる。

## 8. 実験

7 節で実装したプロトタイプを用いて、APVAS+ の性能を実験評価する。従来手法として BGP, BGPsec および APVAS との比較を行う。

### 8.1 実験目的

APVAS+において、経路情報が実際に経路広告された際のメモリ消費量を測定する。具体的には、APVAS+ のメモリ量を従来手法として BGPsec 本来の仕様 [14] 及び APVAS [11] と比較・評価する。これにより、APVAS+の特徴である異なる `Secure_PATH` 同士の署名集約によるメモリ消費量削減の効果を確認する。関連して、APVAS+ が実装された BGP ルータのメモリ消費量が、従来のハイエンドルータの性能である 2GB の範囲で収まるかを確認する。世界規模のフルルート情報に相当する約 80 万経路の静的経路が経路広告された場合を想定して、APVAS+ の現実世界における有効性を確認する。

次に、部分経路広告機能を使用した際の実行時間を測定する。具体的には、ある AS が所持する経路数に対し、その中で一部の `Secure_PATH` だけ部分検証署名 $E$ を生成し経路広告する。このとき、経路数の割合を変化させることで、実行時間の変化を確認する。これにより、経路広告間隔を表す `MRAI (Minimal Route Advertisement)` の通常設定である 30 秒以内で、どの程度までの部分経路広告が最大で処理可能なのか確認する。

### 8.2 実験環境と設定

APVAS+仕様に変更した BIRD（1.6.0 ver.）を仮想コンテナ・ソフトウェア Docker ver 2.2.0.3[*5] の Ubuntu 16.04 イメージ上（メモリ:2B）に搭載する。これにより、各 Docker コンテナが 1 台の仮想ルータとなる。このとき、1 台の BGP ルータは 1 つの AS とみなし、その仮想ルータを複数台実行することで、ネットワークを構築する。なお、Docker コンテナを立ち上げる端末の環境は CPU : 1.3GHz デュアルコア Intel Core i5、Memory : 8GB、OS : mac OS Catalina ver 10.15.5 である。

経路広告する経路としては、フルルート相当の経路情報 80 万経路をランダムに生成した。まず、複数の静的経路を経路広告するルータを 1 台用意し、そこから連続して 5 台のルータを連結させる。このとき、既存研究 [1], [11] はプロトタイプの動作の安定性のため、最大で 200 件程度の経路件数でしか実験がされていない。このため、本実験においても 200 件の静的経路を経路広告した際のメモリ消費量を測定する。一方、作成した APVAS+のプロトタイプで予備実験を行ったところ、Docker コンテナを立ち上げている mac OS 端末のリソースが原因で、平均 `AS_PATH` 長である 5 台 [27] までの経路広告を考えた場合、 約 1 万経路しか安定して経路広告できなかった。このため、1 万件の経路を経路広告し、そこまでの測定結果を元に APVAS+

[*5] `https://www.docker.com/`

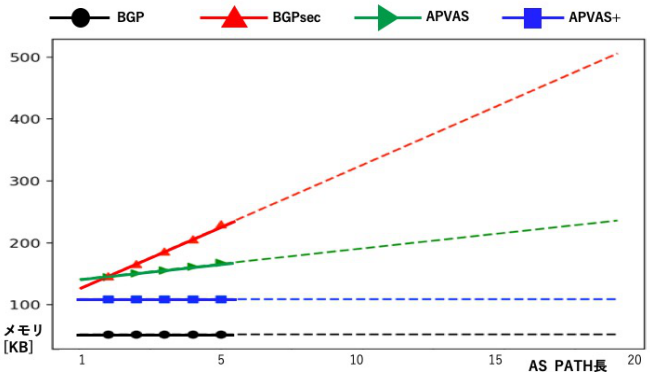


図 3　各プロトコルのメモリ消費量の比較（200 件）

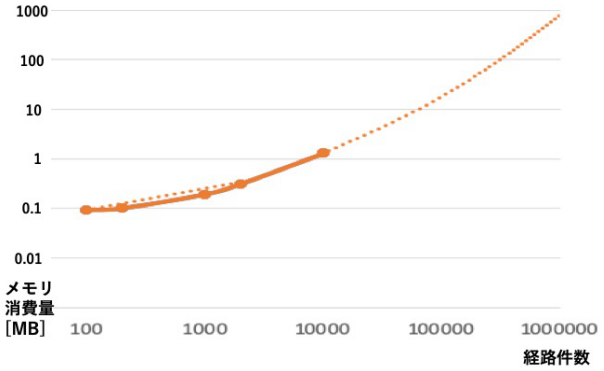


図 4　フルルート規模でのメモリ消費量推定（AS_PATH 長: 5）

における 80 万件でのメモリ消費量を推定した。

このとき、メモリ量を測定する状況としては、経路が収束するまでの最悪時点の数値を利用する。また、BGP において平常時は差分の経路情報を交換しているが、ミスコンフィグレーションや BGP セッションの張り直しの際には大量の経路情報が交換される。その様な最悪時の条件として、約 80 万経路が経路広告される場合を想定してメモリ量の測定を行う。また、本実験に実験に利用するトポロジを構成している各 BGP ルータは経路広告の前にお互いに接続が完了されているものとする。

次に、部分経路広告機能に関しては 2 台の仮想ルータを接続し、送信側が所持する経路数を 500、1000、1500 と変化させる。これらの経路数のうち、相手のルータに経路広告する割合を変化させることで、それぞれの場合で送信側が部分検証用署名を作成する時間と受信側が部分検証を行う時間を測定する。前述したとおり、平均 AS_PATH 長が 5 であることから、2 台の BGP ルータは経路生成元からそれぞれ 4、5 だけ AS_PATH が離れているものとする。

### 8.3　結果

APVAS+と BGP, BGPsec, APVAS におけるメモリ消費量を比較したものを図 3 に示す。APVAS+に関しては経路広告収束前の未送信の署名数が最大となった場合を示している。これにより、異なる経路間の署名を集約できる APVAS+は既存手法よりも署名によるメモリ消費量を削減できていることがわかる。特に、AS_PATH 長が現実的に最大の場合といえる 20 の場合は従来の BGPsec と比較して約 8 割のメモリ量が削減できている。また、フルルート規模でのメモリ消費量の予測結果を図 4 に示す。1 万経路までの実線は実測値であり、波線が推定部分となる。これにより、100 万経路の場合であってもメモリ消費量が 1GB を下回るため、フルルート相当の経路を広告する場合でも現実的なメモリ量で抑えることが可能といえる。

次に、部分検証による実行時間の計測結果を図 5 に示す。これにより、総経路数が 500 件程度までなら 30 秒以内に部分署名検証の生成も検証も行うことができる。

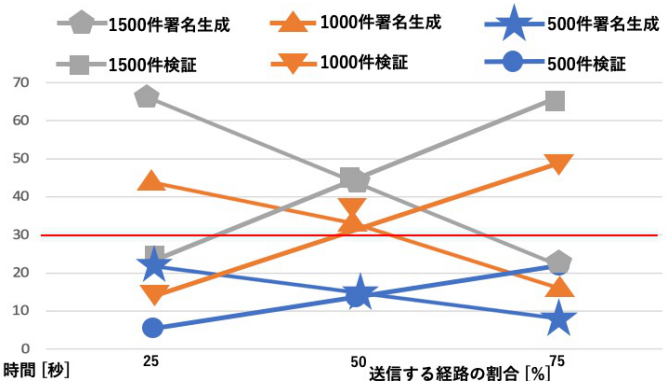


図 5　部分経路広告の実行時間

## 9.　考察

**実験結果に関する考察**：今回は BGP ルータ同士の接続が完了した状態で経路広告を行ったが、接続初期の段階で update メッセージのやりとりにおいて、約 114 経路分（約 7KB）の署名が蓄積されていた。また、経路広告開始後も通信が不安定になった際に、送信待ちのため未集約の署名が蓄積されていた。これらの蓄積の量は高々 4500 個程度（約 290KB）であったことも確認している。

自身が所有する静的経路を経路広告する経路生成元のルータ内では署名を付加して送信するだけで検証は行わないので実行のオーバヘッドが他のルータよりも小さい。そのため、update メッセージを短期間でより大量に送信することが可能である。これより、経路生成元の隣に接続されているルータは受信する update メッセージが過多となり、未集約の署名を蓄積しやすくなる。そのため、送信側が相手ルータの検証時間を考慮して待機時間を設定することで受信側の署名による一時的なメモリ量消費を抑えている。

また、実行時間に関しては大きなオーバーヘッドがあった。具体的には 1 万経路を経路広告する場合、検証と署名生成を行うルータでは実行時間が AS_PATH 長に依存し、AS_PATH 長が 5 の場合は 1 件辺り 12.6 ミリ秒で処理をしている。これは署名生成のみ場合と比べて 6 倍程度の時間になる。実行時間の削減は今後の課題である。

**制約**：今回の実験では SKI を `update` メッセージに含めていない。実際の BGPsec ルータでは SKI も管理されることから、これも含めて実験を行うことが今後の課題である。なお、現状の仮説としては、ルータ内に保存される SKI は AS 台数のみに依存し、異なる `Secure_PATH` 間でも共通であり、個々の SKI のサイズは 20B であることから、概ね 16MB の負荷が積算されるものと予想している。

## 10. まとめ

本稿では BGPsec のメモリ負荷削減を現実的に運用可能なサイズまで実現するプロトコル APVAS+ を提案した。大まかな着想は異なる `Secure_PATH` 間の署名も集約すること及び、集約後も一部の `Secure_PATH` だけを経路広告できる機能を導入したことである。また、この構成のために新しい要素技術として楽観的併存型集約署名も設計した。APVAS+ の性能をフルルート情報相当の経路情報を用いて実験したところ、メモリ負荷が最大で 8 割 まで削減されることで、その有効性を確認した。本実験は SKI を含めておらず、SKI を含めた実験は今後の課題である。

**謝辞**：本研究の一部は JSPS 科研費 18K18049 およびセコム財団挑戦的研究助成の助成を受けたものです。

### 参考文献

[1] Bird bgpsec. `http://www.securerouting.net/tools/bird/`.

[2] G. Asharov, D. Demmler, M. Schapira, T. Schneider, G. Segev, S. Shenker, and M. Zohner. Privacy-preserving interdomain routing at internet scale. *Proceedings on Privacy Enhancing Technologies*, 2017(3):147 – 167, 2017.

[3] H. Birge-Lee, L. Wang, J. Rexford, and P. Mittal. Sico: Surgical interception attacks by manipulating bgp communities. In *Proc. of CCS 2019*, page 431–448. ACM, 2019.

[4] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Proc. of EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 416–432. Springer, 2003.

[5] K. Brogle, S. Goldberg, and L. Reyzin. Sequential aggregate signatures with lazy verification from trapdoor permutations - (extended abstract). In *Proc. of ASIACRYPT 2012*, volume 7658 of *LNCS*, pages 644–662. Springer, 2012.

[6] T. Chung, E. Aben, T. Bruijnzeels, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, R. v. Rijswijk-Deij, J. Rula, and N. Sullivan. Rpki is coming of age: A longitudinal study of rpki deployment and invalid route origins. In *Proc. of IMC 2019*, page 406–419. ACM, 2019.

[7] P. Ekparinya, V. Gramoli, and G. Jourjon. The attack of the clones against proof-of-authority. In *Proc. of NDSS 2020*. Internet Society, 2020.

[8] V. Giotsas, G. Smaragdakis, C. Dietzel, P. Richter, A. Feldmann, and A. Berger. Inferring bgp blackholing activity in the internet. In *Proc. of IMC 2017*, page 1–14. ACM, 2017.

[9] S. Goldberg, M. Schapira, P. Hummon, and J. Rexford. How secure are secure interdomain routing protocols. In *Proc. of SIGCOMM 2010*, page 87–98. ACM, 2010.

[10] G. Hartung, B. Kaidel, A. Koch, J. Koch, and A. Rupp. Fault-tolerant aggregate signatures. In *Proc. of PKC 2016*, volume 9614 of *LNCS*, pages 331–356. Springer, 2016.

[11] O. Junjie, N. Yanai, T. Takemura, M. Okada, S. Okamura, and J. P. Cruz. Apvas: Reducing memory size of as_path validation by using aggregate signatures, 2020.

[12] S. Kent, C. Lynn, and K. Seo. Secure border gateway protocol (S-BGP). *IEEE Journal on Selected areas in Communications*, 18(4):582–592, 2000.

[13] M. Lepinski and S. Kent. *An Infrastructure to Support Secure Internet Routing*. 2012. Published: RFC 6480.

[14] M. Lepinski and K. Sriram. *BGPsec Protocol Specification*. 2017. Published: RFC 8205.

[15] S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures and multisignatures without random oracle. In *Proc. of EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 465–485. Springer, 2006.

[16] R. Lychev, S. Goldberg, and M. Schapira. Bgp security in partial deployment: Is the juice worth the squeeze? *SIGCOMM Computer Communication Review*, 43(4):171–182, 2013.

[17] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *Proc. of EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 74–90. Springer, 2004.

[18] L. Miller and C. Pelsser. A taxonomy of attacks using bgp blackholing. In *Proc. of ESORICS 2019*, volume 11735 of *LNCS*, pages 107–127. Springer, 2019.

[19] M. Nawrocki, J. Blendin, C. Dietzel, T. C. Schmidt, and M. Wählisch. Down the black hole: Dismantling operational practices of bgp blackholing at ixps. In *Proc. of IMC 2019*, page 435–448. ACM, 2019.

[20] Y. Rekhter, S. Hares, and T. Li. *A Border Gateway Protocol 4 (BGP-4)*. 2006. Published: RFC 4271.

[21] J. M. Smith, K. Birkeland, T. McDaniel, and M. Schuchard. Withdrawing the bgp re-routing curtain: Understanding the security impact of bgp poisoning through real-world measurements. In *Proc. of NDSS 2020*. Internet Society, 2020.

[22] J. M. Smith and M. Schuchard. Routing around congestion: Defeating ddos attacks and adverse network conditions via reactive bgp routing. In *Proc. of IEEE S&P 2018*, pages 599–617, 2018.

[23] K. Sriram. RIB Size Estimation for BGPSEC, 2011. `https://www.nist.gov/document-7096`.

[24] K. Tanaka, N. Yanai, M. Okada, T. Nishide, and E. Okamoto. APAT: An Application of Aggregate Signatures to BGPSEC. In *Fast Abstract in DSN 2016*, 2016.

[25] M. Tran, I. Choi, G. J. Moon, A. V. Vu, and M. S. Kang. A stealthier partitioning attack against bitcoin peer-to-peer network. In *Proc. of IEEE S&P 2020*, pages 496–511. IEEE, 2020.

[26] P. Vervier, O. Thonnard, and M. Dacier. Mind your blocks: On the stealthiness of malicious BGP hijacks. In *Proc. of NDSS 2015*. Internet Society, 2015.

[27] C. Wang, Z. Li, X. Huang, and P. Zhang. Inferring the average as path length of the internet. In *Proc. of ICNIDC*, pages 391–395. IEEE, 2016.

[28] M. Zhao, S. W. Smith, and D. M. Nicol. Aggregated path authentication for efficient BGP security. In *Proc. of CCS 2005*, pages 128–138. ACM, 2005.