DeBGP: Decentralized and Efficient BGP Hijacking Prevention System

Jiang Li*, Jiahao Cao*†§, Yuan Yang*§, Zhuotao Liu†§,
Qi Li†§, Yangyang Wang†§, Zili Meng†, Renjie Xie†, and Mingwei Xu*†‡§¶‖
* Department of Computer Science and Technology, Tsinghua University, Beijing, China
† Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing, China
‡ Beijing National Research Center for Information Science and Technology, Beijing, China
§ Zhongguancun Laboratory, Beijing, China
¶ Quan Cheng Laboratory, Shandong, China
‖Peng Cheng Laboratory, Shenzhen, China
{lijiang17@mails, zhuotaoliu@, caojh2021@, yangyuan_thu@mail,
qli01@, wangyy-13@, xrj21@mails, xumw@}.tsinghua.edu.cn, zilim@ieee.org

***Abstract*—BGP is the de facto inter-domain routing protocol that plays an important role on the Internet. However, BGP suffers from the origin and path hijacking due to its lack of validation for BGP update messages. Traditional security proposals protect BGP from the threats based on a centralized authority, i.e., RPKI, in which a single point failure may cause a large-scale failure. In this paper, we propose a decentralized and efficient BGP hijacking prevention system named *DeBGP*. It converts each BGP update message into corresponding blockchain transactions that ASes can query for validation. To cope with the poor transaction processing capability of blockchains, DeBGP groups ASes on the Internet into different consortiums. Multiple local blockchains and collaborative blockchains are crafted to handle BGP validation among consortiums. Moreover, DeBGP enforces local validation within consortiums and blockchain-based validation across consortiums to effectively reduce validation delay. We implement the DeBGP prototype, and experiments with real-world BGP data demonstrate its effectiveness and efficiency on preventing BGP hijacking.**

***Index Terms*—Inter-domain routing, security, decentralization, blockchain**

I. Introduction

Border Gateway Protocol (BGP) [1] is the de facto inter-domain routing protocol on the Internet. Autonomous Systems (ASes) exchange reachability information with BGP update messages. However, the lack of BGP validation for update messages intrinsically causes two types of BGP hijacking, i.e., origin and path hijacking. Origin hijacking involves an attacker announcing the victim's prefix or sub-prefix as its own to divert the victim's traffic to itself. A notable example of this attack is when Pakistan Telecom's misuse of its sub-prefix caused YouTube to be unavailable for more than 2 hours [2]. Path hijacking entails an attacker advertising a shorter fake path to the victim to lure the victim's traffic to itself. A recent example of this attack is when Zayo's spoofing of a shorter path led to the theft of 1.9 million dollars worth of digital assets from cryptocurrency platform KLAYswap [3].

Many solutions [4–11] have been proposed to prevent BGP origin and path hijacking. They validate BGP update messages based on Resource Public Key Infrastructure (RPKI) [12], which is a centralized architecture standardized by Internet Engineering Task Force (IETF). RPKI provides mappings between IP prefix and ASes under the authority of five Regional Internet Registries (RIRs). According to the mappings, ASes can perform origin validation to prevent origin hijacking [4], or performs hop-by-hop path validation cryptographically to prevent path hijacking [13]. However, a single point failure of the centralized authority in RPKI may result in a massive breakdown of the entire architecture [14–16]. Although soBGP [17] and IRV [18] adopt a decentralized architecture, they fail to completely prevent the BGP hijacking. In soBGP, how to establish trust anchors for validating routing information is unclear [19]. IRV may fail to prevent anomalous BGP update messages due to inaccurate validation of update messages [19].

Recently, the decentralized blockchain technology inspires innovation in the network security domain [20, 21]. Thus, an intuitive idea is to incorporate the blockchain technology into BGP message validation to defend against single point failures [22]. ASes on the Internet record the mappings between IP prefix and ASes on a blockchain for origin validation. Each update message in BGP is converted into a corresponding blockchain transaction that other ASes can query for path validation. Nevertheless, blockchain is notorious for its poor scalability. Recent studies [23, 24] show that even the advanced blockchain systems can only process thousands of transactions per second. However, the Internet exchanges millions of BGP update messages per second [25], which can generate millions of relevant blockchain transactions per second. It is far beyond the processing capability of a global blockchain. Furthermore, converting all BGP update messages into blockchain transactions for validation introduces high delay due to the tremendous number of blockchain operations. This can remarkably increase BGP convergence time [23, 26].

In this paper, we present DeBGP that prevents BGP origin and path hijacking in a *decentralized* and *efficient* manner. To cope with the poor transaction processing capability of blockchain, DeBGP groups ASes on the Internet into *consortiums*. Each consortium maintains a local blockchain to convert

BGP update messages originating from it into transactions for validation. Every two adjacent consortiums maintain a collaborative blockchain to convert BGP update messages across them into transactions for validation. Such a design imposes a much lower throughput requirement for each blockchain.

To further reduce the BGP validation delay due to blockchain operations, DeBGP adopts local validation within a consortium and blockchain-based validation across consortiums. Each AS pulls related validation information from the local blockchain in advance, such as the mappings between IP prefix and ASes, and store them locally. For each BGP update message originating from its consortium, an AS can directly perform origin validation and hop-by-hop BGP path validation without querying its local blockchain. As an AS cannot directly pull related validation information on other consortiums, BGP update messages across consortiums are validated by querying the corresponding transactions in collaborative blockchains.

Based on the above design, the security and performance of DeBGP heavily depend on how we group ASes on the Internet into consortiums. Inappropriate grouping can cause the security degradation of DeBGP. For example, if a consortium consists of many ASes that have a high probability of announcing anomalous update messages, the anomalous update messages may get the consortium's endorsement and propagate to ASes in other consortiums, causing the origin or path hijacking. Moreover, inappropriate grouping may incur high BGP message validation latency. If a consortium owns many AS links to other consortiums, there can be a vast number of BGP update messages propagating to different consortiums per second. DeBGP has to frequently record and query the corresponding endorsement transactions in the blockchains across consortiums, which is time-consuming.

To solve the above problem, we present a security score model for the consortium and formalize it as a maximum security grouping (MAX-SEC) problem with performance constraint. We prove the MAX-SEC problem is NP-hard and design a heuristic grouping algorithm taking security and performance aspects into consideration. Our algorithm selects a seed AS as the initial AS for each consortium, and extends consortiums from these initial ASes's neighbors iteratively. Both seed AS selection and consortium extension follow the greedy principle that the candidate AS brings the most security gain and the least cost.

We demonstrate that DeBGP can effectively protect BGP from the origin and path hijacking through security analysis. We implement the DeBGP prototype and conduct extensive experiments with real BGP update message data on the real Internet AS topology. The results show that a single consortium's failure in DeBGP only cause less than 4% global influence in terms of the allocation of about 4 billion IPv4 addresses and 70,000 ASes on the Internet. Compared to the standardized and centralized BGP security proposal named *BGPsec*, DeBGP reduces 21% validation cost and 31% validation time with about 16 million paths from real AS routing tables in the global Internet. The convergence time of DeBGP is no more than 0.3% longer than that of BGPsec with various experimental topologies.

Our main contributions are summarized as follows.

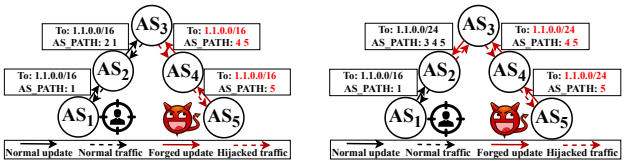
- We propose a decentralized and efficient BGP origin and path hijacking prevention system named DeBGP.
- We formalize the AS grouping problem in DeBGP, prove its NP-hardness, and design a heuristic algorithm to solve it.
- We design the intra-consortium local validation and inter-consortium blockchain-based validation mechanisms to effectively reduce the validation cost.
- We conduct extensive experiments with real BGP data to prove the effectiveness and efficiency of DeBGP.

II. Background

Border Gateway Protocol (BGP). BGP [1] is a policy-based routing protocol to exchange reachability information among Autonomous Systems (ASes). The update message BGP uses contains the IP prefixes that origin AS originates and some path attributes. The AS path along which receiver ASes can reach the prefixes is the most important path attribute. However, BGP has no built-in mechanism to validate BGP update messages, therefore any AS can announce arbitrary update messages, resulting in two types of hijacking described below.

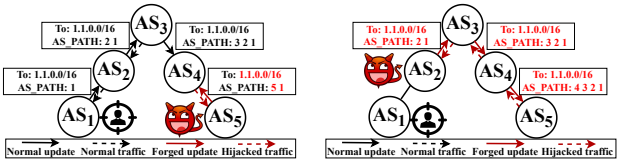
- **Origin hijacking**: a malicious AS falsely announces the victim AS's prefix or sub-prefix to attract traffic. For example, in Fig. 1(a), AS_5 forges the victim AS_1's prefix. In this case, the traffic from AS_4 to AS_1 will be directed to AS_5 according to the shortest-path principle. The traffic from AS_3 to AS_1 may be directed to AS_5. Fig. 1(b) shows a more powerful attack, the malicious AS_5 forges a more specific prefix of the victim AS_1's prefix, which causes the traffic from AS_2, AS_3, and AS_4 to be directed to the AS_5 according to the longest prefix match principle.
- **Path hijacking**: There are two types of path hijacking. In the first type, i.e., path manipulation attack, a malicious AS usually forges a shorter path to attract traffic. As shown in Fig. 2(a), AS_5 forges a link with the victim AS_1, causing the traffic from AS_4 to be directed to AS_5 according to the shortest-path principle. In the second type, i.e., unauthorized route announcement, a malicious AS announces an update message unauthorized from the victim AS. As shown in Fig. 2(b), although there exists a link between the malicious AS_2 and the victim AS_1, AS_1 does not announce an update message to AS_2, but AS_2 announces to AS_3 that it can reach to a particular prefix of the victim AS_1. This violates the victim AS_1's intent.

As mentioned before, BGP is a policy-based routing protocol, therefore, BGP update message propagation follows the Gao-Rexford routing principle [27] based on the AS business relationship. AS business relationships fall into two broad categories: customer-provider (abbreviated as c2p or p2c) and settlement-free peering (p2p). In the c2p (or p2c) relationship, the provider AS and its customer AS announce update



(a) Prefix hijacking. (b) Sub-prefix hijacking.

Fig. 1: Origin hijacking.



(a) Path manipulation attack. (b) Unauthorized route announcement.

Fig. 2: Path hijacking.

messages destined to all the prefixes on the Internet in all. In the p2p relationship, the two ASes announce update messages destined to their and their customers' prefixes to each other.

Blockchain. The participants in the blockchain system form a decentralized network. Participants use multi-party consensus mechanisms such as Proof of Working (PoW) to maintain a distributed ledger recording transactions and smart contracts in a hash chain for tamper-proof property. Transaction data shows the interaction between the participants. Smart contracts are programs that execute agreements negotiated by participants automatically and securely.

III. Design Overview

DeBGP aims to protect ASes from BGP origin and path hijacking in a divide-and-conquer manner. The architecture of DeBGP is shown in Fig. 3. The limitation of blockchain scalability poses a significant challenge for its application in validation of tremendous BGP messages. DeBGP groups ASes on the Internet into **consortiums** to improve scalability. ASes in the consortium maintain *INRchain* to record the mappings between IP prefix and ASes, and *PathKeychain* to record AS public keys for path validation. *TrustRelaychain* is maintained between adjacent consortiums to record endorsements of update messages across consortiums.

Based on the above blockchains, DeBGP adopts hop-by-hop validation in consortiums and blockchain based validation across consortiums. Intra-consortium mechanism validates update messages in the consortium as follows. ASes perform origin validation with local prefix to AS mappings from *INRchain*. For path validation in the consortium, ASes validate the AS path hop by hop with public keys of corresponding ASes from the *PathKeychain*. Inter-consortium mechanism validates update messages across consortiums as follows. Update message sender AS records the corresponding transaction as endorsement from its consortium for the update message across consortiums on the *TrustRelaychain*. Receiver ASes validate the update message by querying the corresponding endorsement transaction. Furthermore, consortium public keys are also recorded on the *TrustRelaychain*. At last, benign update messages are propagated in and across consortiums

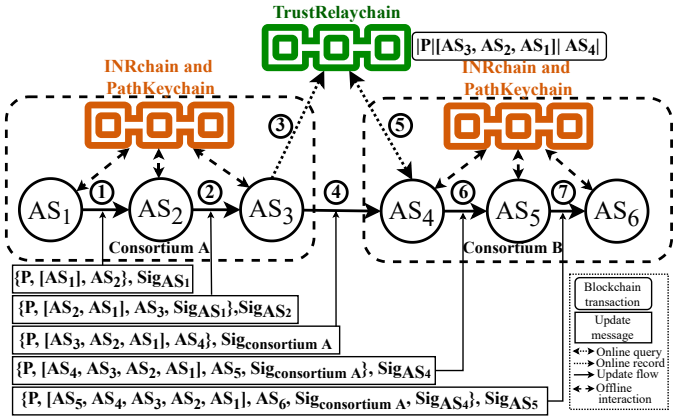


Fig. 3: DeBGP†.

† **Data in {} is the object to be signed.**

and the anomalous ones are prevented by DeBGP.

In the Fig. 3 example, consortium A consists of AS_1, AS_2, and AS_3, while consortium B consists of AS_4, AS_5, and AS_6. AS_1 originates an update message with prefix $\mathcal{P}$ propagating from it to AS_6. To protect BGP, DeBGP workflow is divided into three steps: validation in consortium A, validation across consortium A and B as well as validation in consortium B.

For validation in consortium A, AS_1 signs the prefix ($\mathcal{P}$), AS path (AS_1), and target AS (AS_2). AS_2 performs origin validation against the update message from AS_1 with the prefix to AS mapping from *INRchain* in consortium A. For path validation, AS_2 verifies what AS_1 signs with AS_1's public key from *PathKeychain* in consortium A. Then, AS_2 signs the prefix ($\mathcal{P}$), AS path ([AS_2,AS_1]), target AS (AS_3) and preceding signature (Sig_{AS_1}). AS_3 performs similar validation as AS_2 against the update message from AS_2.

For validation across consortium A and B, before AS_3 announces the update message to AS_4, AS_3 records the corresponding endorsement transaction of the update message on the *TrustRelaychain*. Then, AS_3 removes AS_1's and AS_2's signatures and signs the prefix ($\mathcal{P}$), AS path ([AS_3, AS_2, AS_1]) and target AS (AS_4) with the consortium A's private key. AS_4 queries the corresponding endorsement transaction for the update message from AS_3 on the *TrustRelaychain*. AS_4 accepts the update message if the corresponding endorsement transaction exists, or else it rejects the update message.

For validation in consortium B, AS_4 signs the update message like above and sends it to AS_5. AS_5 verifies AS_4's signature with AS_4's public key from *PathKeychain* in consortium B and consortium A's signature with consortium A's public key from the *TrustRelaychain*. Then, AS_5 signs the update message, and AS_6 performs validation as above.

IV. Grouping ASes on Internet into Consortiums

Internet is a small-world network with high clustering coefficient [28], which lays the foundation for our grouping. We first present the consortium security score model, and then formulate a maximum security grouping (MAX-SEC) problem. At last, we analyze the MAX-SEC problem and develop an algorithm for the problem.

TABLE I: Notation Table

Symbol	Definition
$S(g_u)$	Security score of consortium g_u
$se(ij)$	Security score of edge (i,j), $se(ij)=sn(i)+sn(j),\forall e_{ij}=1$
$sn(i)$	Security reputation of AS i, $i\in V, sn(i)>0$
e_{ij}	Binary indicator of existence of edge (i,j) in G, =1 if edge (i,j) exists in G or 0 otherwise
l_{ij}	Binary indicator, =1 if AS_i and AS_j belong to the same consortium and 0 otherwise. $l_{ij}=\sum_{u\in\{1,2...K\}} b_{iu}b_{ju}$
α	Weight for the internal edge of consortium
β	Weight for the crossing edge of consortium
$G(V,E)$	Internet graph
K	A $k-way$ grouping of G, pairwise disjoint subsets of G
g_u	K consortiums of G after grouping, $\bigcup g_u=G$, $\forall u\in\{1,2...K\}, g_v\bigcap g_w=\varnothing \quad \forall v,w\in\{1,2...K\}$
b_{iu}	Decision variable, =1 if AS i is in the consortium g_u otherwise 0
$C(g_u)$	=1 if consortium g_u is connected in topology or 0 otherwise
$\|g_u\|$	Size of consortium g_u, the number of ASes in the consortium g_u
$Minsize$	Minimum size of consortium
$Maxsize$	Maximum size of consortium
$Degree(i)$	Degree of AS i
$CondRatio$	Threshold of the consortium conductance, which is the ratio between the number of crossing edges and the sum of degrees of all the ASes in it
$PRatio$	Threshold of the ratio between the number of p2p crossing edges and the number of all crossing edges of the consortium
r_{ij}	Binary indicator of edge (i,j) business relationship type, =1 if AS_i is AS_j's provider (or customer) or 0 if AS_i and AS_j are peers of each other

A. Consortium Security Score Model

The consortium security is influenced by the following factors. First, the security reputation of ASes that make up the consortium influences the consortium security. For example, a consortium consisting of ASes that often announce anomalous update messages [29, 30] would not be trustworthy by other consortiums. We get AS security reputation from existing AS reputation system [31]. Second, the number of ASes in a consortium influences the consortium security. A consortium with few ASes is prone to error. Third, the location of ASes in a consortium matters. The announcement behavior of border ASes influences at least two consortiums.

Considering all factors above, we define the consortium security score as Eq. (1). The consortium security score is expressed by the security score of AS edge, which is determined by security reputations of both endpoint ASes. $\alpha<\beta$ holds between weight for the internal and crossing edge of consortium since we want the crossing edge which crosses two consortiums to have high security score. We summarize our symbols and their definitions in Table I.

$$S(g_u)=\sum_{i\vee j\in g_u} se(ij)e_{ij}[\beta(1-l_{ij})+\alpha l_{ij}] \quad \forall i,j \tag{1}$$

B. Problem Formulation

The Internet is modeled as an undirected graph $G(V,E)$ with node set V and edge set E. V is the set of ASes on Internet, and E is the set of edges between ASes. We group ASes on Internet into K consortiums. Let Φ be the minimum security score of all consortiums. We aim to maximize Φ to improve overall security of all consortiums, since each consortium is the trust anchor in DeBGP. Moreover, we consider scalability and cost constraint. The maximum security grouping (MAX-SEC) problem is as follows.

$$\max\Phi \tag{2}$$

$$s.t. \quad \Phi\le S(g_u) \quad \forall u\in\{1,2\cdots K\} \tag{3}$$

$$\sum_{u\in\{1,2\cdots K\}} b_{iu}=1 \quad \forall i\in V \tag{4}$$

$$C(g_u)=1 \quad \forall u \tag{5}$$

$$Minsize\le|g_u|\le Maxsize \quad \forall u \tag{6}$$

$$\frac{\sum_{i\in g_u}Degree(i)-2\sum_{i\wedge j\in g_u}e_{ij}}{\sum_{i\in g_u}Degree(i)}\le CondRatio \quad \forall u \tag{7}$$

$$\frac{\sum_{i\vee j\in g_u}r_{ij}(1-l_{ij})}{\sum_{i\vee j\in g_u}(1-l_{ij})}\le PRatio \quad \forall e_{ij}=1 \quad \forall u \tag{8}$$

Eq. (3) means that Φ is the minimum security score of all consortiums. we maximize the minimum security score instead of the average score of all consortiums due to the power law distribution property of Internet [32]. Eq. (4) means the grouping is not overlapping. Eq. (5) guarantees that each consortium g_u is topologically connected. Eq. (6) means that the size of the consortium must not be less than $Minsize$ and greater than $Maxsize$. As the number of participants in a blockchain system increases, the transaction throughput (i.e., transactions per second) decreases. The number of ASes in a consortium influences the consortium security. For example, a consortium with only 3 ASes is vulnerable to failure. If an AS malfunctions, the other two ASes may not reach a consensus and cause the consortium to malfunction as well.

Eq. (7) illustrates that the conductance of a consortium should be lower than $CondRatio$, i.e., the number of consortium crossing edges should be fewer. With Eq. (6) and Eq. (7), a consortium has high security score not due to numerous ASes and crossing edges. Eq. (8) specifies that crossing edges of the consortium should be p2p relationship as far as possible. The number of update messages on p2p edges is always fewer than that on c2p (or p2c) edges. Update messages on p2p edges indicate the two peer ASes' and their customers' prefixes. Yet, the update messages on c2p (or p2c) edges indicate all the prefixes on the Internet.

C. Problem Analysis

Theorem 1. *The MAX-SEC problem is NP-hard.*

We prove the theorem by a polynomial time reduction from the Max-Cut Problem, which is NP-hard [33]. For a graph, a maximum cut is a cut whose size is at least the size of any other cut. That is, it is a partition of the graph's vertices into two complementary sets S and T, such that the number of edges between the set S and the set T is as large as possible. The problem of finding a maximum cut in a graph is known as the Max-Cut Problem [34]. We give detailed proof as follows.

Proof: For each instance of the Max-Cut problem in graph, i.e., graph $G_0(V_0,E_0)$, we will construct an instance of P1 problem. Fig. 4 shows an example. Graph $G(V,E)$ is first constructed as $G_0(V_0,E_0)$. Weight of edges in E_0 are all assigned 1. Then we add two nodes r_1 and r_2 to V_0 to construct V. The two nodes only have edges of weight 0 with all nodes in V_0. Let α be 0, β be 1 and K be 2. For Eq. (6) we set MinSize as 0 and MaxSize as $|V|$. For Eq. (7) and Eq. (8), we set $CondiRatio$ and $PRatio$ as 1. The construction

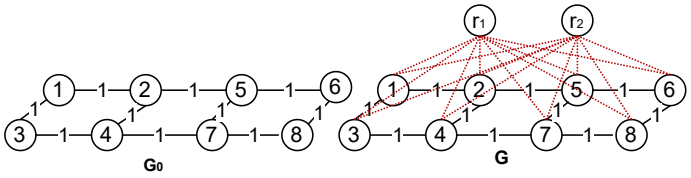


Fig. 4: An example for proof of Theorem 1

of the above instance of P1 can be done in polynomial time.

We now show that finding the max cut in G_0 is equivalent to finding the optimal solution of problem P1 in G. On the one hand, given the max cut $C(S,T)$ in G_0, we construct P as $(S\cup r_1,T\cup r_2)$. Due to our construction process above, $S\cup r_1$ is connected, since there is an edge between r_1 and every node in S. Similarly, $T\cup r_2$ is connected, thus P is an available solution of P1. Furthermore, we can see that $S(g_u)$ in our variable calculation equals the total edge weight between $S\cup r_1$ and $T\cup r_2$, which equals the edge number between S and its complement T, since the edge weights between r1(r2) and nodes in T(S) are all 0. Thus, $S(g_u)$ is maximized since $C(S,T)$ has the maximum cut edge number, which means that P is the optimal solution of P1. On the other hand, given the optimal solution of P1, i.e. $P(a_1,a_2)$, we construct a cut C in G_0 as $(a_1\setminus(r_1\cup r_2),a_2\setminus(r_1\cup r_2))$. We prove C is the max cut in G_0 by contradiction. If we find a cut C' bigger than C, then we could construct a solution better than P from C' by the method above, which is a contradiction. Thus, C is the max cut in G_0.

From the process above, we can see that finding the max cut in G_0 is equivalent to finding the optimal solution of problem P1 in G. The construction can be done in polynomial time. This ends our proof. Problem P1 is NP-hard. ■

According to Theorem 1, it is not easy to find the optimal solution of the MAX-SEC problem. Thus, we propose a heuristic algorithm to solve the problem below.

D. Heuristic AS grouping algorithm

Heuristic algorithm of Max-Cut Problem cannot be used for the MAX-SEC problem due to two reasons. First, Max-Cut Problem does not take connectivity into consideration, but the MAX-SEC problem demands connectivity of all consortiums. Second, the MAX-SEC problem demands the number of crossing edges to be less, however, Max-Cut Problem does not have this constraint. Therefore, we need to design a custom algorithm.

The algorithm ranks ASes as candidate seed list in terms of two metrics, i.e., degree and security reputation, with same weight (line input). The algorithm prioritizes ASes with high degree and low security reputation for two reasons. First, it is easier for high degree ASes to expand their neighbors into their consortium. Second, We expect ASes at the border of the consortium to have high security reputation since they record blockchain endorsement transactions for update messages across consortiums. While seed ASes are probably inside the consortium, thus it is better to select seed ASes with low security reputation to leave ASes with high security reputation at the border. At last, the algorithm outputs consortium set (line output).

Algorithm 1: AS grouping algorithm

input : The number of consortiums, K; The seed candidate list of ASes that are ranked by their degree and security reputation, $(a_1,a_2,\cdots,a_n)$

output: The consortium set g, $(g_1,g_2,\cdots,g_K)$.

1 **initialization** $s_1=a_1, S=(s_1), N=\mathcal{N}(s_1), i=1$
2 **while** $|S|<K$ **do**
3 $i\leftarrow i+1$
4 **if** $a_i\in N$ **then**
5 **continue**
6 **else**
7 $S\leftarrow S\cup a_i$
8 $N\leftarrow N\cup\mathcal{N}(a_i)$
9 **for** $i\leftarrow 1$ **to** K **do** // seed set S, $(s_1,s_2,\cdots,s_K)$
10 $g_i.add(s_i)$
11 **while** *AS grouping not over* **do**
12 g.$sort$(ConsortiumSecurityScore)
13 **for** $i\leftarrow 1$ **to** K **do**
14 candidate=$FindBest(g_i)$
15 **if** *Candidate meets constrains* **then**
16 $g_i.add$(candidate)
17 **else**
18 **continue**
19 **return** g

The algorithm sets the first seed s_1 as the first AS in the seed candidate list, i.e., a_1, and adds s_1 in seed set. Then, the algorithm sets the neighbor set of seeds N as the neighbor set of s_1. The function $\mathcal{N}(s_i)$ returns the set of ASes that are the neighbors of s_i. Empirically, the neighbors of s_i consists of ASes three hops away from s_i (line 1). The first loop continues until K seeds are selected (line 2). If the next candidate AS is in the current neighbor set of seeds, the algorithm re-selects next candidate AS (line 3-5). Otherwise, the algorithm updates seed set and their neighbor set (line 6-8). The loop stops when seed set S construction is finished.

After selecting seeds, the algorithm expands consortiums around these seeds. The algorithm first adds each seed to its corresponding consortium (line 9-10). The algorithm continues to add ASes into consortiums until no ASes are left, then the grouping is finished (line 11). The algorithm sorts consortiums in ascending order of security score (line 12). Low security score consortiums have priority to first find best candidate ASes, which conforms to the optimization goal of the MAX-SEC problem. After sorting consortiums, the algorithm finds the candidate AS of consortium g_i through $FindBest(g_i)$ (line 13-14). The function $FindBest(g_i)$ selects the best candidate AS from the consortium g_i's neighbor ASes that have not been grouped into other consortiums. In this way, Eq.(4) and Eq.(5) hold for every consortium. The best candidate AS brings the most security gain to the consortium g_i. It also has the most edges and c2p (or p2c) edges with ASes in the consortium g_i. This corresponds to Eq.(1), Eq. (7) and Eq. (8), respevtively. Then the algorithm checks

TABLE II: *INRchain* Transaction

Transaction Type	Format
INR Registration	$\|INR\|Entity\|$
INR Revocation	$\|INR\|Entity\|$
Origin Authorization Registration	$\|Prefix\|Maxlen\|OriginAS\|$
Origin Authorization Revocation	$\|Prefix\|Maxlen\|OriginAS\|$

whether the candidate AS meets constrains, i.e., Eq. (6), Eq. (7) and Eq. (8) (line 15). $Maxsize$, $Minsize$, $CondRatio$ and $PRatio$ in these equations are set empirically based on security and cost. If the candidate AS meets constrains, the algorithm adds it in the corresponding consortium (line 16) or else the algorithm repeats the process for the next consortium (line 17-18). When all ASes are added in consortiums, the algorithm returns consortium set g (line 19).

V. DeBGP security mechanism with blockchain

After grouping ASes on Internet into consortiums, we design DeBGP intra-consortium and inter-consortium security mechanism with blockchain. Intra-consortium mechanism validates update messages in the consortium, while inter-consortium mechanism validates update messages across consortiums. DeBGP is shown in Fig. 3.

A. Intra-consortium Mechanism

1) Origin Validation: To protect BGP from origin hijacking, the consortium in DeBGP records Internet number resource (INR, including IP prefix and ASN) ownership and prefix to AS mappings on the blockchain, i.e., *INRchain*. For an update message, ASes in the consortium check whether the origin AS is the legitimate originator of the prefix according to the prefix to AS mapping from the *INRchain*.

The *INRchain* manages Internet number resource and prefix to AS mappings in the consortium through four types of transaction, as shown in Table II. INR Registration and Revocation transaction record the INR ownership of entities like ISPs. Origin Authorization Registration and Revocation transaction indicate prefix to AS mappings with the maximum length of the prefix that can be originated. Every transaction also records its initiator and endorsement from other ASes.

INRchain smart contract is executed by ASes in the consortium for transaction validation as Algorithm 2. For the INR Registration transaction, the smart contract checks the INR ownership through multiple data sources, i.e., `Whois` [35] and `PeeringDB` [36] (line 3-4). For the INR Revocation transaction, the smart contract checks whether a corresponding INR Registration transaction exists and the initiators of the two transactions are the same (line 5-7). For the Origin Authorization Registration transaction, the smart contract checks whether the initiator entity of this transaction is the same as the Entity in the INR Registration transaction of the corresponding prefix. Then, it checks the origin authorization using prefix to origin AS mapping in update messages from route collectors of `RIPE RIS` [37] and `Route Views` [38] as evidence (line 8-10). For the Origin Authorization Revocation transaction, the smart contract checks like INR Revocation transaction (line 11-13). If transactions pass the validation above, they will be recorded on the *INRchain*.

Algorithm 2: *INRchain* smart contract

```
1  initialization ASes
     executing the smart contract receive a transaction
2  switch Transaction Type do
3      case INR Registration do
4          OwnershipCheck(INR)
5      case INR Revocation do
6          RegistrationExistence(INR)
7          SameInitiatorCheck(Registration, Revocation)
8      case Origin Authorization Registration do
9          SameEntityCheck(Initiator, INR Registration)
10         MappingCheck(Prefix, Origin AS)
11     case Origin Authorization Revocation do
12         RegistrationExistence(Authorization)
13         SameInitiatorCheck(Registration, Revocation)
```

2) Path Validation: To protect BGP from path hijacking, DeBGP designs a new path attribute DeBGP_SIG in BGP update message to record signatures. Each consortium maintains a blockchain *PathKeychain* to manage AS public keys in a decentralized way. ASes validate the AS path consisting of ASes in the same consortium hop by hop with corresponding AS public keys from the *PathKeychain*. After validation, ASes fill the DeBGP_SIG field with its signature of the prefix, AS path, the target AS in the same consortium and preceding signatures.

B. Inter-consortium Mechanism

We elaborate how to perform origin and path validation for update messages across consortiums after ensuring BGP security in the consortium. In DeBGP, two adjacent consortiums maintain a blockchain *TrustRelaychain* to record endorsement of the update message across the two consortiums and public keys of the two consortiums.

Before ASes announce an update message to ASes in another consortium, they first record the endorsement of the update message from their consortium on the *TrustRelaychain*. The endorsement is an UpdateEndorsement transaction on the *TrustRelaychain*. The UpdateEndorsement transaction contains the prefix, AS path and receiver AS of the update message across consortiums. Its format is $|Prefix|ASPath|TargetAS|$. Then ASes sign the prefix, AS path and target AS as a whole with their consortium's public key. At last, ASes fill the DeBGP_SIG field in the update message with the signature and send the update message to the target AS in another consortium. The target AS verifies the signature with public key of preceding consortium from the *TrustRelaychain*. Moreover, the target AS queries the corresponding UpdateEndorsement transaction of the update message on the *TrustRelaychain*.

TrustRelaychain smart contract is executed by ASes in the consortium for UpdateEndorsement transaction validation as Algorithm 3. The smart contract first extracts the Prefix and ASPath fields from the UpdateEndorsement transaction (line 1). If the length of ASPath is 1, the smart contract checks whether the last AS in the ASPath is the legitimate origin of the Prefix based on the local prefix to AS mappings from

the *INRchain* (line 2-3). For ASPath of length greater than 1, the smart contract checks each AS in the ASPath except ASPAth[0], i.e., transaction initiator (line 4-5). If the AS is in the same consortium with the transaction initiator AS, it checks whether it has announced the update message, which contains the Prefix and AS path from ASPath[-1] to itself, to its target AS based on its Adj-RIBs-Out table (line 6-8). The Adj-RIBs-Out table of AS BGP router contains the routes for advertisement to specific peers [1]. Otherwise, the AS is not in the same consortium with the transaction initiator AS. If the AS is the first AS that is in a different consortium from ASPath[0], the smart contract checks whether the corresponding UpdateEndorsement transaction is on the *TrustRelaychain* maintained with the AS's consortium (line 9-12). Otherwise, the smart contract terminates (line 13-14). If the UpdateEndorsement transaction passes the validation above, it will be recorded on the *TrustRelaychain* maintained with the target AS's consortium. The consortium public key management on the *TrustRelaychain* is omitted due to space limit.

Algorithm 3: *TrustRelaychain* smart contract

```
1  initialization
     Prefix, ASPath=Extract(UpdateEndorsement)
2  if Len(ASPath) == 1 then
3  |  LocalINRchainCheck(Prefix, ASPath[-1])
4  else
5  |  for AS in ASPath[1:] do
6  |  |  if SameConsortium(AS, ASPath[0]) then
7  |  |  |  i ← ASPath.index(AS)
8  |  |  |  AdjRIBsOutCheck(Prefix,
   |  |  |    ASPath[i:-1], ASPath[i-1])
9  |  |  else
10 |  |  |  if FirstDifferentConsortium(AS,
   |  |  |    ASPath[0]) then
11 |  |  |  |  i ← ASPath.index(AS)
12 |  |  |  |  TrustRelaychainCheck(Prefix,
   |  |  |  |    ASPath[i:-1], ASPath[i-1])
13 |  |  |  else
14 |  |  |  |  break
```

VI. Security analysis

A. Trust Model

There are two trust models now: "root of trust" and "web of trust". The former has the drawback of centralization due to a single root trust anchor. The latter does not specify the criteria for establishing trust, like the number of trust anchors. DeBGP uses a variant of "web of trust" model [39], i.e., "consortium of trust" model. In the model, every consortium after our grouping algorithm is a trust anchor. In DeBGP, ASes in the consortium reach consensus on regulation of their routing behaviors. When ASes announce update messages to ASes in different consortiums, their consortium endorses the update messages. The endorsement is recorded on the blockchain for query from other ASes. In this way, trust propagates along AS path through consortiums and tamper-proof blockchains maintained between them.

B. Security Effectiveness Analysis

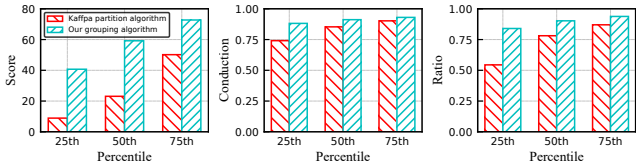
Protection from origin hijacking. In DeBGP, an arbitrary AS can prevent prefix/sub-prefix hijacking aiming at other ASes in the same consortium. For update messages across consortiums, sender consortium endorses them to guarantee security. In the Fig. 1(a) example, we suppose AS_4 and the malicious AS_5 are in different consortiums. AS_5 tries to hijack the victim AS_1's prefix, and the consortium of AS_5 will not endorse the corresponding anomalous update message from AS_5. AS_5 can still announce the anomalous update message to AS_4 even if its consortium does not endorse the message. AS_4 will reject the anomalous update message since it cannot find the corresponding endorsement transaction for the update message on the *TrustRelaychain* maintained between its consortium and AS_5's consortium. The analysis is similar for sub-prefix hijacking in Fig. 1(b).

Protection from path hijacking. The path is secure in each consortium due to hop-by-hop validation in DeBGP. For the update message across consortiums, the sender AS signs the path with its consortium's private key. To prevent anomaly of the sender AS, sender AS's consortium endorses the update message. The corresponding blockchain endorsement transaction is recorded on the *TrustRelaychain* maintained between the sender consortium and the receiver consortium. The receiver AS validates the update message by querying the corresponding transaction. In the Fig. 2(a) example, we suppose AS_4 and the malicious AS_5 are in different consortiums. AS_5 forges the link with the victim AS_1 in an anomalous update message, and the consortium of AS_5 will not endorse the update message since *AdjRIBssOutCheck()* or *TrustRelaychainCheck()* in Algorithm 3 will prevent the corresponding UpdateEndorsement transaction. AS_5 can still announce the anomalous update message to AS_4 even if its consortium does not endorse the message. AS_4 will reject the anomalous update message due to similar reason as above. The analysis is similar for unauthorized route announcement in Fig. 2(b).

VII. Evaluation

A. Experiment Setup

We implement DeBGP prototype with `Hyperledger Fabric` [23] for blockchains and `ExaBGP` [40] as the agent to interact with BGP daemons and blockchains. `FRR` [41] and `FRR-based BGPsec` [42] are used for BGP and BGPsec daemon, respectively. `CAIDA` [43] datasets are used for evaluation of our grouping algorithm and system decentralization. The datasets contain real Internet AS topology with business relationship as well as the allocation of about 4 billion IPv4 addresses and 70 thousand ASNs. We collect about 16 million paths from real ASes' routing tables and update message stream of 15 minutes from `Route Views` [38] for evaluation of path validation cost and router CPU utilization. `SimBGP` [44] is used to evaluate the convergence time in common experimental topologies. All our experiments are conducted on Dell PowerEdge R740 Rack Server with Intel(R)



(a) Security score. (b) Conductance. (c) P2p crossing edges ratio.

Fig. 5: Grouping evaluation

Xeon(R) Gold 6230R CPU @ 2.10 GHz and 128 GB RAM.

B. Grouping Evaluation

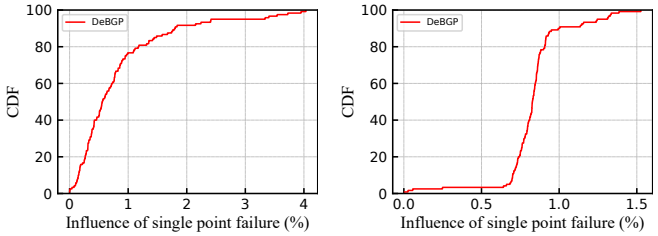
The consortium security score indicates how secure the consortium is. As Fig. 5(a) shows, DeBGP outperforms `Karlsruhe Fast Flow Partitioner (Kaffpaa)` [45], one of the most popular graph partitioning tools, in the 25th, 50th, and 75th percentile of consortiums. DeBGP realizes a 186% consortium security score improvement than Kaffpaa on average. Particularly, DeBGP improves security scores of low score consortiums more than high score consortiums compared to Kaffpaa since low security score consortiums have priority to first find best candidate ASe in DeBGP.

The consortium conductance defined in Eq. (7) reflects the consortium's degree of coupling with its neighbors. A consortium with fewer AS links to its neighbors has smaller conductance and thus lower inter-consortium interaction cost. As Fig. 5(b) shows, DeBGP achieves similar conductance as Kaffpaa, whose optimization objective is to minimize the conductance. DeBGP degrades about 10% consortium conductance than Kaffpaa on average. Both algorithms realize relatively high conductance, since the Internet is a well-connected network.

We use the ratio between the number of p2p crossing edges and the number of all crossing edges of the consortium to evaluate cost. A consortium with the high ratio always processes fewer update messages across consortiums. As Fig. 5(c) shows, the ratio of DeBGP is 26% higher than Kaffpaa on average. With the increase of the percentile, the ratio of Kaffpaa is approaching DeBGP, since the Internet is transiting from a transit hierarchy to a peering mesh [46].

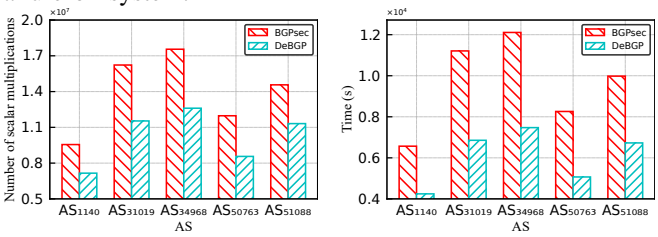
C. Performance Evaluation

1) Decentralization: The decentralization of DeBGP is evaluated by the influence of a single point failure on the whole system. We analyze how much IPv4 address space and how many ASes on the Internet will be influenced if a consortium fails. As Fig. 6(a) shows, more than 90% single consortium failures in DeBGP cause less than 2% influence in terms of IPv4 address space. Moreover, all single consortium failures cause less than 4% influence. Fig. 6(b) shows that more than 90% single consortium failures in DeBGP cause less than 1% influence in terms of ASes. Moreover, all single consortium failures cause less than 2% influence. Nevertheless, BGPsec relies on the centralized RPKI under the authority of 5 RIRs. Any RIR's failure would cause severe global breakdown, for example, American Registry for Internet Numbers (ARIN) failure would cause influence on



(a) IPv4 address space. (b) AS.

Fig. 6: Decentralization in terms of influence of single point failure on system.



(a) Number of the scalar multiplications. (b) Time of cryptographic computations.

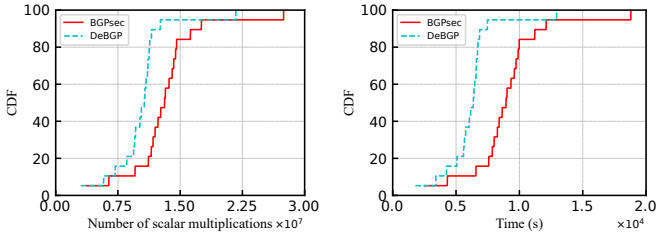
Fig. 7: Path validation cost for some ASes.

about 46% IPv4 address space on the Internet.

2) Path Validation Cost: Cryptographic computations incur the most cost of path validation, which include the generation and verification of signatures. The Elliptic Curve Digital Signature Algorithm (ECDSA) with curve P-256 is used for signing and verifying update messages in both BGPsec [13] and DeBGP. The ECDSA signature verification is considerably slower than generation. The most computationally intensive operation of ECDSA, i.e., the scalar multiplication, is required twice for signature verification but only once for signature generation [47, 48].

Fig. 7(a) shows the number of scalar multiplications for all ASes along the paths in some ASes's routing tables. DeBGP reduces 26% number of the scalar multiplications than BGPsec in path validation. For example, to construct a path [1140, 6939, 4788, 15932, 9587, 24378] in AS_{1140}'s routing table, all 6 ASes along the path perform 5 signature generations and 15 signature verifications in BGPsec. Cryptographic computations in BGPsec require 35 scalar multiplications in total. In DeBGP, our AS grouping algorithm puts AS_{1140}, AS_{6939}, AS_{4788} in one consortium and AS_{15932}, AS_{9587}, AS_{24378} in another consortium. All 6 ASes along the path perform 7 signature generations (additional 2 signature generations for the *TrustRelaychain* endorsement transaction) and 9 signature verifications. These cryptographic computations require 25 scalar multiplications in total. Fig. 7(b) shows the cryptographic computation time for path validation in our experiment setup. DeBGP reduces 37% cryptographic computation time than BGPsec in path validation.

We use AS paths from routing tables of monitor ASes from `Route Views` [38] AMS-IX Collector to evaluate path validation cost for them. Fig. 8(a) and Fig. 8(b) show the distribution of the number of the scalar multiplications and the time of cryptographic computations of these monitor ASes, respectively. DeBGP reduces 21% number and 31%



(a) Distribution of number. (b) Distribution of time.

Fig. 8: Distribution of path validation cost.

TABLE III: Convergence time (s)

Topology	BGPsec	DeBGP
B-Clique-6	138.19	138.29
Clique-6	86.26	86.35
CRYSTAL-5	86.26	86.35
Focus-16	86.52	86.71
Grid-16	136.28	136.38
Ring-15	169.14	169.64

time of cryptographic computations than BGPsec.

3) CPU Utilization: BGP processes consume the majority of CPU cycles on the router [49], thus router CPU load is a significant concern for operators. Fig. 9 shows the CPU utilization of BGPsec and DeBGP. For monitor ASes from AMS-IX Collector of `Route Views` [38], we feed their corresponding real BGP update message stream of 15 minutes to a BGP daemon for CPU utilization measurement. On average, DeBGP reduces 51% router CPU utilization than BGPsec.

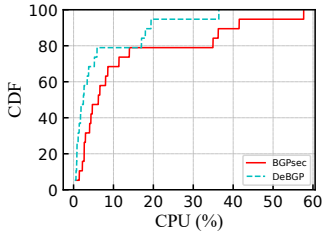


Fig. 9: CPU utilization.

4) Convergence Time: Table III shows convergence time of BGPsec and DeBGP in different topologies [26]. Update message processing delay of an AS is required for convergence time evaluation of `SimBGP`. We get the processing delay of BGPsec and DeBGP from emulation experiment. Table III indicates that DeBGP increases the convergence time by no more 0.3% than BGPsec. The transaction recording and querying of blockchain introduces delays in update message validation, but this causes little impact on convergence time.

VIII. Related Work

Centralized BGP security architecture. RPKI [12] and BGPsec [13] validate BGP update messages based on a centralized authority. Any single point failure of the authority may cause a massive global breakdown. In addition, hop-by-hop path validation in BGPsec incurs massive repeated signature verification cost. Both of them are inspired by S-BGP [4]. Other variants including psBGP [50], SPV [5], APA [10], Hu et al.'s cumulative authentication mechanism and FS-BGP [11] try to reduce centralization or path validation cost, but they all still rely on a centralized authority. psBGP [50] uses a rating mechanism similar to that used by PGP [39] for prefix origination and BGPsec-like way for path validation. However, psBGP assumes the existence of a centralized trust model for AS number management. SPV (Secure Path Vector) [5] explores symmetric cryptographic techniques to perform path validation. Aggregated path authentication (APA) uses signature amortization and aggregate signatures to reduce the path validation cost [7–9]. Hu et al. proposes to use message authentication code (MAC) to speed up the path validation [10]. FS-BGP [11] realizes a similar security level as BGPsec through signing critical AS-path segments.

Decentralized BGP security architecture. soBGP [17] avoids the reliance on the centralized authority. However, it leaves open the issue of how to establish trust anchors for validation of the signed objects [19]. Moreover, soBGP cannot prevent BGP hijacking comprehensively. soBGP cannot verify that AS path in the update message is consistent with what the preceding ASes along the path announce, which BGPsec can. The validation of update messages may fail in IRV [18] since the response of AS IRV servers to queries about their routes may be wrong [19]. Additionally, some studies propose blockchain-based BGP security solutions [22, 51, 52], however, they focus less on blockchain scalability challenges and BGP path validation.

BGP anomaly detection. Some studies such as PHAS [53], ARTEMIS [54], Argus [55], Themis [56] and Dong et al.'s proposal [57] perform detection after BGP anomalies occur. They process collected control plane data and data plane probing data with various algorithms to detect BGP anomalies. Different from the above existing studies, our work prevents the anomalies in advance.

IX. Conclusion

We propose DeBGP, a decentralized and efficient BGP hijacking prevention system. DeBGP converts each BGP update message into corresponding blockchain transactions that ASes can query for validation. To resolve the scalability issues incurred by blockchains, we group ASes on the Internet into different consortiums taking security and validation cost into consideration. We present the intra-consortium mechanism to guarantee BGP security within a consortium, and the inter-consortium mechanism to relay trust between consortiums. We implement the DeBGP prototype and analyze its security effectiveness in preventing BGP hijacking. Our experiments demonstrate the effectiveness and efficiency of DeBGP.

Acknowledgment

The research is supported in part by the National Natural Science Foundation of China (NSFC) under Grant 62221003, 61832013, 62132011, 62132004 and 62202260; and in part by the China Postdoctoral Science Foundation under Grant 2022M721824; and in part by the Shuimu Tsinghua Scholar Program. Mingwei Xu is the corresponding author of the paper.

References

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

[2] "youtube," https://www.ripe.net/publications/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study.

[3] C. Cimpanu, "Klayswap crypto users lose funds after bgp hijack - the record by recorded future," https://therecord.media/klayswap-crypto-users-lose-funds-after-bgp-hijack/.

[4] S. Kent, C. Lynn, and K. Seo, "Secure border gateway protocol (s-bgp)," *IEEE JSAC*, 2000.

[5] Y.-C. Hu, A. Perrig, and M. Sirbu, "Spv: Secure path vector routing for securing bgp," in *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, 2004, pp. 179–192.

[6] B. Raghavan, S. Panjwani, and A. Mityagin, "Analysis of the spv secure routing protocol: Weaknesses and lessons," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 2, pp. 29–38, 2007.

[7] D. M. Nicol, S. W. Smith, and M. Zhao, "Efficient security for bgp route announcements," 2003.

[8] M. Zhao, S. W. Smith, and D. M. Nicol, "Aggregated path authentication for efficient bgp security," in *Proceedings of the 12th ACM conference on Computer and communications security*, 2005, pp. 128–138.

[9] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Advances in Cryptology—EUROCRYPT 2003: International Conference on the Theory and Applications of Cryptographic Techniques, Warsaw, Poland, May 4–8, 2003 Proceedings 22*. Springer, 2003, pp. 416–432.

[10] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Efficient security mechanisms for routing protocolsa." in *Ndss*. Citeseer, 2003.

[11] Y. Xiang, X. Shi, J. Wu, Z. Wang, and X. Yin, "Sign what you really care about–secure bgp as-paths efficiently," *Computer Networks*, vol. 57, no. 10, pp. 2250–2265, 2013.

[12] M. Lepinski and S. Kent, "An Infrastructure to Support Secure Internet Routing," IETF RFC 6480.

[13] M. Lepinski and K. Sriram, "BGPsec Protocol Specification," RFC 8205.

[14] B. Rothenberger, D. E. Asoni, D. Barrera, and A. Perrig, "Internet kill switches demystified," in *Proceedings of the 10th European Workshop on Systems Security*, 2017, pp. 1–6.

[15] K. Shrishak and H. Shulman, "Limiting the power of rpki authorities," in *Proceedings of the Applied Networking Research Workshop*, 2020, pp. 12–18.

[16] D. Cooper, E. Heilman, K. Brogle, L. Reyzin, and S. Goldberg, "On the risk of misbehaving rpki authorities," in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, 2013, pp. 1–7.

[17] R. White, "Securing bgp through secure origin bgp (sobgp)," *Business Communications Review*, vol. 33, no. 5, pp. 47–53, 2003.

[18] G. Goodell, W. Aiello, T. Griffin, J. Ioannidis, P. D. McDaniel, and A. D. Rubin, "Working around bgp: an incremental approach to improving security and accuracy in interdomain routing." in *NDSS*, vol. 23. Citeseer, 2003, p. 156.

[19] G. Huston, M. Rossi, and G. Armitage, "Securing bgp—a literature survey," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 2, pp. 199–222, 2010.

[20] S. Matsumoto and R. M. Reischuk, "Ikp: turning a pki around with decentralized automated incentives," in *Proc. IEEE S&P*, 2017.

[21] E. Karaarslan and E. Adiguzel, "Blockchain based dns and pki solutions," *IEEE Communications Standards Magazine*, 2018.

[22] A. Hari and T. Lakshman, "The internet blockchain: A distributed, tamper-resistant transaction framework for the internet," in *Proceedings of the 15th ACM workshop on hot topics in networks*, 2016, pp. 204–210.

[23] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proc. EuroSys*, 2018.

[24] C. Li, P. Li, D. Zhou, Z. Yang, M. Wu, G. Yang, W. Xu, F. Long, and A. C.-C. Yao, "A decentralized blockchain with high throughput and fast confirmation," in *2020 {USENIX} Annual Technical Conference ({USENIX}{ATC} 20)*, 2020, pp. 515–528.

[25] G. Huston, "Bgp in 2021 — bgp updates — apnic blog," https://blog.apnic.net/2022/01/13/bgp-updates-2021/.

[26] S. Deshpande and B. Sikdar, "On the impact of route processing and mrai timers on bgp convergence times," in *IEEE Global Telecommunications Conference, 2004. GLOBECOM'04.*, vol. 2. IEEE, 2004, pp. 1147–1151.

[27] L. Gao, "On inferring autonomous system relationships in the internet," *IEEE/ACM Transactions on networking*, vol. 9, no. 6, pp. 733–745, 2001.

[28] X. Wang and D. Loguinov, "Understanding and modeling the internet topology: economics and evolution perspective," *IEEE/ACM Transactions on Networking*, vol. 18, no. 1, pp. 257–270, 2009.

[29] M. Konte, R. Perdisci, and N. Feamster, "Aswatch: An as reputation system to expose bulletproof hosting ases," in *Proc. ACM SIGCOMM*, 2015.

[30] C. Testart, P. Richter, A. King, A. Dainotti, and D. Clark, "Profiling bgp serial hijackers: capturing persistent misbehavior in the global routing table," in *Proc. ACM IMC*, 2019.

[31] "bgpranking," https://bgpranking-ng.circl.lu/.

[32] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," *ACM SIGCOMM computer communication review*, vol. 29, no. 4, pp. 251–262, 1999.

[33] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.

[34] Wikipedia contributors, "Maximum cut - wikipedia," https://en.wikipedia.org/wiki/Maximum_cut.

[35] "Whois," https://who.is/.

[36] "Peeringdb," https://www.peeringdb.com/.

[37] "Ripe ris (routing information service)," https://www.ripe.net/analyse/internet-measurements/routing-information-service-ris/ris-raw-data.

[38] "Routeviews," http://www.routeviews.org/routeviews/.

[39] P. R. Zimmermann, *The official PGP user's guide*. MIT press, 1995.

[40] "Exa-networks/exabgp: The bgp swiss army knife of networking," https://github.com/Exa-Networks/exabgp.

[41] "Frrouting," https://frrouting.org/.

[42] T. C. Schmidt, "Implementation and evaluation of bgpsec for the frrouting suite."

[43] "Caida." https://www.caida.org/.

[44] "Simbgp: Python event-driven bgp simulator," http://www.bgpvista.com/simbgp.php.

[45] P. Sanders and C. Schulz, "Kahip v3. 00–karlsruhe high quality partitioning–user guide," *arXiv preprint arXiv:1311.1714*, 2013.

[46] A. Dhamdhere and C. Dovrolis, "The internet is flat: modeling the transition from a transit hierarchy to a peering mesh," in *Proc. ACM CoNEXT*, 2010.

[47] A. Antipa, D. Brown, R. Gallant, R. Lambert, R. Struik, and S. Vanstone, "Accelerated verification of ecdsa signatures," in *Selected Areas in Cryptography: 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005, Revised Selected Papers 12*. Springer, 2006, pp. 307–318.

[48] J. Petit, "Analysis of ecdsa authentication processing in vanets," in *2009 3rd International Conference on New Technologies, Mobility and Security*. IEEE, 2009, pp. 1–5.

[49] S. Agarwal, C.-N. Chuah, S. Bhattacharyya, and C. Diot, "Impact of bgp dynamics on router cpu utilization," in *Proc. PAM*. Springer, 2004.

[50] T. Wan, E. Kranakis, and P. C. van Oorschot, "Pretty secure bgp, psbgp." in *NDSS*. Citeseer, 2005.

[51] Q. Xing, B. Wang, and X. Wang, "Bgpcoin: Blockchain-based internet number resource authority and bgp security solution," *Symmetry*, vol. 10, no. 9, p. 408, 2018.

[52] J. Paillisse, M. Ferriol, E. Garcia, H. Latif, C. Piris, A. Lopez, B. Kuerbis, A. Rodriguez-Natal, V. Ermagan, F. Maino *et al.*, "Ipchain: Securing ip prefix allocation and delegation with blockchain," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 2018, pp. 1236–1243.

[53] M. Lad, D. Massey, D. Pei, Y. Wu, B. Zhang, and L. Zhang, "Phas: A prefix hijack alert system." in *USENIX Security Symposium*, 2006.

[54] P. Sermpezis, V. Kotronis, P. Gigis, X. Dimitropoulos, D. Cicalese, A. King, and A. Dainotti, "Artemis: Neutralizing bgp hijacking within a minute," *IEEE/ACM Transactions on Networking*, 2018.

[55] X. Shi, Y. Xiang, Z. Wang, X. Yin, and J. Wu, "Detecting prefix hijackings in the internet with argus," in *Proc. ACM IMC*, 2012.

[56] L. Qin, D. Li, R. Li, and K. Wang, "Themis: Accelerating the detection of route origin hijacking by distinguishing legitimate and illegitimate {MOAS}," in *31st USENIX Security Symposium (USENIX Security 22)*, 2022, pp. 4509–4524.

[57] Y. Dong, Q. Li, R. O. Sinnott, Y. Jiang, and S. Xia, "Isp self-operated bgp anomaly detection based on weakly supervised learning," in *2021 IEEE 29th International Conference on Network Protocols (ICNP)*. IEEE, 2021, pp. 1–11.