The Horizontal INR Conflict-Detection Algorithm: Revealing INR Reallocation and Reauthorization in RPKI*

Hui Zou
Computer Network Information Center, Chinese Academy of Sciences
University of Chinese Academy of Sciences
ZDNS Corporation
Beijing, China
zouhui@cnic.cn

Di Ma, Qing Shao, Wei Mao
Computer Network Information Center,
Chinese Academy of Sciences
ZDNS Corporation
Beijing, China
madi, shaoqing, mao@zdns.cn

***Abstract*—Resource Public Key Infrastructure (RPKI) is a promising security enhancement to the Border Gateway Protocol, but it only requires the relying party (RP) to validate Internet Number Resource (INR) allocation or authorization relationships expressed in parent-child certificate pairs vertically. Therefore, conflicts in INR allocation and authorization may exist because of the limitations of the validation procedure of the RP software, in other words, certification authority malfunctions in issuing RPKI objects within a publication point cannot be detected by the RP. We develop a model of such conflicts and propose a horizontal INR conflict-detection algorithm with acceptable build time and query time. The proposed algorithm was tested on real-world RPKI data to identify actual and potential INR conflicts and its accurateness has been tried to be evaluated. This paper also discusses the deployment issues and the accuracy dependence about our algorithm design.**

***Index Terms*—BGP, RPKI, INR conflict**

I. Introduction

Although it is the de-facto inter-domain routing protocol, the Border Gateway Protocol (BGP) was designed without considering security. Any autonomous system (AS) can hijack destinations by advertising IP prefixes it does not control, resulting in prefix and sub-prefix hijacks. The Resource Public Key Infrastructure (RPKI) [1] provides a trusted mapping from Internet Number Resources (INRs), such as IP prefixes and autonomous system numbers (ASNs), to their holder by binding a public key and these resources in a resource certificate (RC). Thus, INR holders can make verifiable statements to confirm that they are authorizing autonomous systems (ASes) to originate routes for IP prefixes in BGP by binding them in route origin authorizations (ROAs), thereby preventing such attacks.

RPKI mirrors the IP address allocation hierarchy. Regional Internet Registries (RIRs) are trust anchors (TAs), allocating IP prefixes to Local Internet Registries (LIRs) or Internet Service Providers (ISPs). The LIRs or ISPs then allocate sub-prefixes to other ISPs or customers, who can then further authorize them to ASes. Accordingly, the relying party (RP) software validates INR-encompassing relationships between a certification authority (CA) and its subordinate CAs and/or issued ROAs [2] [3], which requires that a CA can only allocate or authorize INRs allocated by its superior CA.

While RPKI eliminates risks for inter-domain routing system due to the lack of security mechanisms in BGP, it may introduce new ones from CA malfunctions in ROAs and RCs. Previous studies [4] [5] [6] have highlighted that it is technically feasible for a CA to allocate the same INRs to different customers (*INR reallocation*), allocate unauthorized INRs to customers (*illegally INR allocation*), and authorize INRs that are reserved or already allocated (*INR reauthorization*). To this end, Liu et al. [7] proposed and implemented a pre-control mechanism to ensure the security and accuracy of resource allocation information. In particular, as the scope of certification of each TA expands from current holding resources to all resources, the risk of cross-RIR INR reallocation also increases. CA malfunctions will lead to INR conflicts [8], making prefix hijackings legitimate and networks unreachable, thereby adversely affecting BGP routing. However, the fact that the INR allocation relationships expressed in RCs are not used to guide inter-domain routing directly and the AS0 mechanism is not widely deployed, which together result in these potential conflicts not being taken seriously.

With the proposal of different types of RPKI-based applications (e.g. Resource Tagged Attestations (RTA) [9]) and the support of the AS0 mechanism by trust anchors, the problems caused by INR conflicts will become increasingly prominent and will no longer be limited to the inter-domain routing system. In addition, real-time detection and early-warning mechanism are of great significance to avoid attacks. However, except for unauthorized INR allocation, other INR conflicts cannot be detected by RPs. This limitation can be attributed to the fact that RP only validates INR encompassing relationships for parent-child certificate pairs vertically, which is referred to as the vertical INR conflict-detection algorithm (*vertical algorithm*) in this study.

To mitigate risks from RPKI, an RP should be able to detect

978-3-903176-32-4 © 2021 IFIP

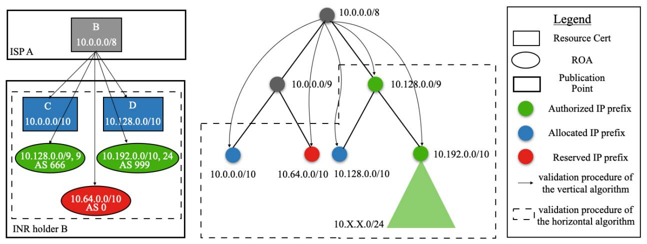

Fig. 1. Vertical Algorithm vs. Horizontal Algorithm

all CA malfunctions in RCs and ROAs. To this end, we propose a horizontal INR conflict-detection algorithm (*horizontal algorithm*), which is the supplement to the vertical algorithm. The horizontal algorithm aims to detect INR reallocations and INR reauthorizations under the premise of legal INR allocations, which is guaranteed by the vertical algorithm. Fig. 1 depicts the differences between the vertical algorithm and the horizontal algorithm, where the former checks INR encompassing relationship for each parent-child certificate pair (e.g. RC B and RC C, RC B and an EE certificate embedded in the AS0 ROA) to reveal illegal INR allocation and authorization (INR conflict A), and the latter checks the existence of the last five types of INR conflicts defined in Section 2 between RCs and ROAs issued by the same INR holder (e.g. all RCs and ROAs issued by RC B). In summary, the main contributions of this study are as follows:

- We first introduce three basic concepts based on the INR usage specification: *authorizable prefix set*, *allocatable prefix set*, and *non-routed prefix set*, and also define rules for them.
- We develop the INR conflict model, which contains six different types of INR conflicts covering all possible CA malfunctions. These INR conflicts are divided into two categories: INR allocation conflicts and INR authorization conflicts.
- We propose the horizontal INR conflict-detection algorithm, which works as a supplement to the vertical algorithm, to detect more INR conflicts. This study is the first of its kind to reveal CA malfunctions by adopting an "after-control" mechanism.
- Numerical results demonstrate the feasibility of our proposed algorithm, as the build time and query time are within an acceptable range. Moreover, four types of INR conflicts (three actual and one potential) are revealed in the current RPKI deployment.

II. The INR Conflicts

A. Basic Concepts

The entire IP address space can be organized into a binary tree, in which each non-leaf node has two child nodes, indicating that each prefix has two child prefixes. Each leaf node represents a prefix that contains a single IP address. As shown in Fig. 2(a), the child prefixes of P are p_{12} and p_{34}; accordingly, P is the parent of p_{12} and p_{34}. Similarly, the child prefixes of p_{12} are p_1 and p_2 , and the child prefixes of p_{34} are p_3 and p_4. Furthermore, p_1, p_2, p_3, p_4, p_{12}, and p_{34} are collectively defined as sub-prefixes of P. In this study, the set of all sub-prefixes of P is defined as $Prefix_{SET}(P)$. Thus, in the example in Fig. 2, p_1, p_2, p_3, p_4, p_{12}, p_{34}, $P \in Prefix_{SET}(P)$. In addition, **ASN** represents all autonomous system numbers, and all IP prefixes can be written as **PREFIX**.

The following example is used throughout this study. We suppose that the INR holder (Alice) has obtained INRs from an ISP or an RIR, and divides it into four disjoint sub-spaces p_1, p_2, p_3, and p_4, where $p_1 \cup p_2 \cup p_3 \cup p_4 = P$, and $p_1 \cap p_2 \cap p_3 \cap p4 = \emptyset$. The different usages of these sub-spaces together define the usage strategy for P, where p_1 is used for numbering hosts of her network, p_2 is reserved for future use, and p_3 and p_4 are allocated to her customers.

1) INR Usage Specification: For each IP address space identified by an IP prefix, there are three usage scenarios as follows:

1) Alice uses it herself;
2) Alice reserves it for future use;
3) Alice allocates it to her customers.

Furthermore, Alice has corresponding operations in RPKI as follows:

1) issuing an ASX ROA (X $\neq$ 0) for INRs she uses herself and certifying "right of use" of these INRs;
2) issuing an AS0 ROA for INRs reserved for future use, indicating these INRs should not be advertised in the inter-domain routing system;
3) issuing RCs for INRs allocated to her customers and certifying ownership of INRs.

The INR holder can apply one or more of these scenarios to the entire address space or sub-spaces of the IP prefix.

2) Authorizable Prefix Set: It should be noted that there is no strict correspondence between holding IP prefixes and advertising IP prefixes. As shown in Fig. 2(b), in addition to the IP prefix P, Alice can also advertise its sub-prefixes, including p_{12}, p_{34}, p_1, and all sub-prefixes of p_1. Sub-prefixes she uses herself can be authorized to ASX(X$\neq$0) by issuing ROAs to originate in BGP. This can be described as: $AU_{SET}(P) = Prefix_{SET}(P) - Prefix_{SET}(p_2) - Prefix_{SET}(p_3) - Prefix_{SET}(p_4)$, where $AU_{SET(P)}$ is the authorizable prefix set of P.

3) Allocatable Prefix Set: According to the above example, we know that p_3, p_4, and all their sub-prefixes have been allocated to Alice's customers. How to use the sub-prefixes of p_3 and p_4 is determined by the customers themselves, and Alice must not advertise them for her own use. For each IP prefix P owned by the INR holder, the sub-prefixes allocated to her customers are defined as elements of the allocatable prefix set $AL_{SET}(P)$.

4) Non-routed Prefix Set: AS0 ROA is a mechanism for INR holders to express their negative intent toward what should not be seen and accepted in the context of BGP routing. Alice can issue an AS0 ROA for her reserved IP prefix p_2 to lock on it. Thus, BGP Speakers can reject BGP announcements whose IP prefix equals p_2 if AS0 is the only origin AS for it. For each IP prefix P owned by the INR holder, those sub-prefixes reserved for future use belong to the non-routed prefix set $NR_{SET}(P)$.

5) Rules:

- For any IP prefix P owned by the INR holder, $AU_{SET}(P) \cup AL_{SET}(P) \cup NR_{SET}(P) = Prefix_{SET}(P)$ and $AU_{SET}(P) \cap AL_{SET}(P) \cap NR_{SET}(P) = \emptyset$.
- Any IP prefix $p \in AU_{SET}(P)$ can be authorized to multiple ASes but not to AS0.
- Any IP prefix $p \in AL_{SET}(P)$ can be allocated to only one customer.
- Any IP prefix $p \in NR_{SET}(P)$ can only be authorized to AS0.

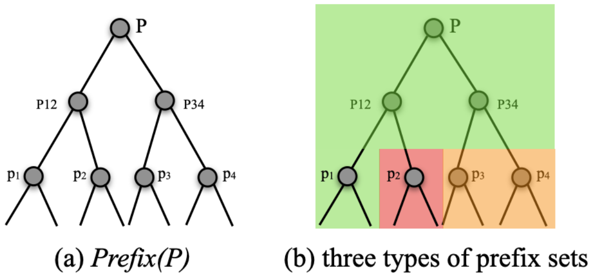

Fig. 2. *Prefix(P)* and three types of prefix sets

B. INR Allocation Conflict

A CA may accidentally or maliciously allocate the same set of IP prefixes to different customers or allocate INRs it does not own. The feasibility of such INR allocation conflicts was previously confirmed by [4]. It is important to note that RPKI follows the "make-before-break" rule to ensure the effective transfer of IP addresses, and this type of INR conflict is regarded as legal. With the increase of TA certificates in INRs, the risk of cross-RIR INR allocation conflicts also increases. When an INR allocation conflict occurs among INR holders (not RIRs themselves) belonging to different RIRs, it can be confirmed that the cross-RIR INR allocation conflict has occurred according to the top-down allocation structure of IP prefixes. To summarize, the three types of INR allocation conflicts, referred to as "INR reallocations" in this study, are:

1) *INR conflict A*: The CA allocates/authorizes INRs that are not allocated by its superior CA to its customers/ROAs. This may result in conflicting INRs in two or more RCs whose issuers are different;
2) *INR conflict B*: Two or more TAs allocate the same INRs to their customers. This results in conflicting INRs in RCs whose issuers are different;
3) *INR conflict C*: The CA which is not a TA allocates the same INRs to its customers. This results in conflicting INRs in RCs whose issuers are the same.

Although conflict A is included in the INR conflict model, it is not within the detection scope of the horizontal algorithm. The reason is that this type of conflict can be filtered out by the vertical algorithm, ensuring that RCs and ROAs that dissatisfy the INR encompassing relationship are not taken as inputs of the horizontal algorithm.

C. INR Authorization Conflict

Owing to the requirement of fine-grained traffic flow control, RPKI provides a way for a CA to authorize an IP prefix and its more specific prefixes together in a ROA using the maxLength attribute. In addition, multiple ROAs can exist with different origin ASes for the same set of IP prefixes to effectively implement traffic engineering, multi-homing, and resource transfer. Ideally, INR holders only authorize IP prefixes in its authorizable prefix sets. However, in practice, maxLength misconfigurations [10] could result in INR conflicts in the form of reauthorization of INRs.

AS0 ROA [11] is a type of ROA with the AS value equaling zero. For an AS0 ROA, prefixes described in this ROA, and any more specific prefixes, should not be used in inter-domain routing system. RPKI allows concurrent issuance of AS0 ROAs and ASX ROAs(X$\neq$0) for the same set of IP prefixes, and the former is used as a default setting to "lock" IP prefixes to prevent them from being illegally broadcast by attackers. However, opposite routing intents for the same set of INRs may lead to potential inconsistency in the validity of BGP announcements.

Up to now, APNIC [12], RIPE NCC [13] and AFRINIC [14] have published AS0 ROAs covering the undelegated IPv4 and IPv6 spaces under their management, and supported the AS0 ROA creation service for their customers. With the global deployment of RPKI, the number of AS0 ROAs and the types of RP softwares will increase, therefore, the differences in the specific implementation and the network performance of RP softwares will lead to potential conflicts. Suppose there exist an AS0 ROA and an ASX ROA(X$\neq$0) for the IP prefix p. We can assume a situation where RP A only downloads the AS0 ROA and RP B has ASX ROA(X$\neq$0) or both because of differences in network performance. RP A and RP B will have different validity states for BGP announcements that should be validated against these ROAs. To summarize, there are three types of INR authorization conflicts, referred to as "INR reauthorizations" in this study:

1) *INR conflict D*: The CA allocates an IP prefix to its customer and also authorizes it to ASX(X$\neq$0) for its own use. This results in conflicting INRs in an RC and an ASX ROA(X$\neq$0);
2) *INR conflict E*: The CA reserves an IP prefix for future use but also authorizes it to ASX(X$\neq$0) for its own use. This results in conflicting INRs in an AS0 ROA and an ASX ROA(X$\neq$0);
3) *INR conflict F*: The CA reserves an IP prefix for future use but also allocates it to its customer. This results in conflicting INRs in an AS0 ROA and an RC.

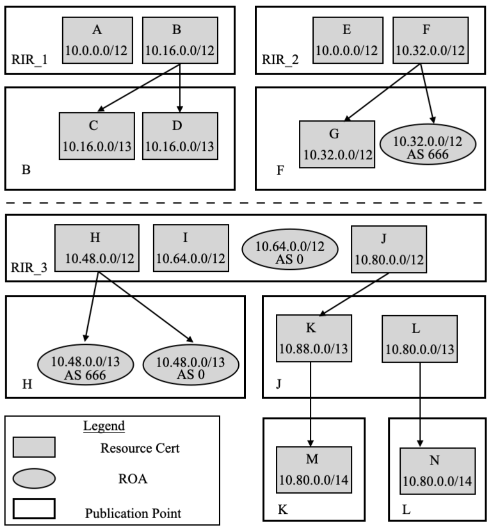

Fig. 3. INR Conflict Model

III. Horizontal INR Conflict Detection Algrithom

The horizontal algorithm can be divided into two modules: *prefix tree construction* and *INR conflict detection*. To accelerate the deployment of RPKI, some operations do not conform to relevant specifications, e.g., RIRs provide the hosted service for their members to help them create and store ROAs, and "responsible grandparenting" [15] allows INR holders to form a relationship with their grandparents for assistance in issuing RCs or maintaining publication points. Therefore, we mainly determine INR allocation relationships based on their hierarchical structure, instead of subject key information (SKI) and authority key information (AKI) matching.

1) *Prefix tree construction*. The prefix allocation tree is a binary tree with left and right branches representing 0 and 1 character respectively, and all characters on the path from the root node to a specific node together constitute the bit string of an IP prefix. All valid RCs and ROAs that have passed the vertical algorithm are inputs of the horizontal algorithm and IP prefixes extracted from valid RCs are added in a prefix allocation tree based on *trie* data structure, using the bit string of an IP prefix as key and related information about a RC that contains this prefix as values, e.g. SKI and AKI fields. The node also stores VRPs (Validated ROA Payloads) [8], which are represented as a set of tuples (ASN, ROA prefix, prefix length, max length) and generated from valid ROAs. These VRPs meet the condition that "ROA prefix/prefix length" in it should be equal to or is the sub-prefix of the prefix represented by this node. Normally, only one RC is stored in a node, indicating that an IP prefix only belongs to only one INR holder.
2) *INR conflicts detection*. The horizontal algorithm aims to detect all types of INR conflicts except *INR conflict A* defined in the INR conflict model. For a given IP prefix, the horizontal algorithm first finds the RC it belongs to. Then, it finds all RCs and ROAs issued by this RC and classifies these objects into three categories: AS0 ROAs, ASX ROAs(X$\neq$0), and RCs. We assume that all three types of these RPKI objects exist. If there are two or more RCs, we can determine the type of INR reallocation conflict by comparing their AKI fields. If their AKI fields are same, INR conflict C can be determined, namely the INR holder issues two or more RCs for the same IP prefix, e.g. RC C and RC D. If their AKI fields are different, INR conflict B can be determined, meaning cross-RIR INR reallocation must occur, e.g. RC A and RC E. Two or more conflicting RIRs can be found by tracing upward in the way of SKI and AKI fields matching, e.g. RIR_1 and RIR_2. The INR reauthorization conflict is determined by checking whether AS0/ASX(X$\neq$0) ROA pairs and ROA/RC pairs contain conflicting INRs. If there exists one AS0 ROA and ASX ROA(X$\neq$0) pair containing the same IP prefix, INR conflict E can be determined, e.g. ROAs issued by RC H. If an ASX ROA(X$\neq$0) and a RC or an AS0 ROA and a RC has the same IP prefix, INR conflict D (e.g. RC G and ROA issued by RC F) or INR conflict F (RC I and ROA issued by RIR_3) can be determined.

In particular, detection of potential INR conflict D is also carried out. For INR holders who have issued ROAs and whose customers may not apply for RCs at present, the customers getting RCs may lead to INR conflict D. We begin by finding invalid BGP announcements validated against ROAs and select those due to AS mismatch. If there exists the provider-customer relationship according to CAIDA's AS relationship dataset [16] between the AS in an invalid BGP announcement and the AS in the ROA validates this BGP announcement, the potential existence of INR conflict D can be confirmed. That is to say, the INR holder has issued a ROA containing IP prefixes allocated to its customer with its own ASN instead of the customer's ASN.

IV. Evaluation and Analysis

In this section, we evaluate the performance of the horizontal algorithm in terms of build time and query time in the first. Our results indicate that the proposed horizontal algorithm has fast build time and query time, and both are within the acceptable range. Then, we introduce the characteristics of RPKI system and illustrate the limitations of the horizontal algorithm. Finally, we analyze false positives.

A. Experimental Setup

The horizontal algorithm was implemented in the Python language and tested on a PC with Intel Core i7 2.2 GHz CPU and 16 GB DDR3 SDRAM. We collected all RPKI objects using the RPstir2 [17] software, and BGP announcements from all BGP collectors were provided by RouteViews [18] and the RIPE routing information service [19] on April 14, 2020. The results indicated that 17,173 valid RCs had passed the vertical algorithm and the number of ROAs was 31,790.

These valid RPKI objects were used as our experimental dataset. Because our work is the first of its kind to reveal INR conflicts results from CA malfunctions by adopting an “after-control” mechanism, there is no other similar existing works to compare, thus we designed two groups of experiments for comparison. The control group did not use any data structure, that is, for any INR holder, the brute-force method was used to find out all RCs and ROAs issued by it and INR conflicts are revealed based on rules in Section 2. The study group is the implementation of our proposed algorithm.

B. Build Time

Before detecting INR conflicts, our proposed algorithm must first build up a prefix allocation tree, constructing each node and inserting related RCs and ROAs into its place in the tree. To estimate the build time, which is the execution time of insertion, 95,061 IP prefixes of all valid RCs were imported. To avoid insertion conflicts, we built two prefix allocation trees, one is for IPv4 prefixes and the other is for IPv6 prefixes. The results show that the average build time for the prefix allocation tree of our proposed algorithm tends to be approximately 466.91s ($\approx$8min). It should be noted that the prefix allocation tree can be built only once during initialization and subsequent modification operation can be realized by a combination of query and insert operations.

C. Query Time

There are two usage scenarios for detecting INR conflicts: 1) reveal INR conflicts for a given IP prefix, and 2) reveal all INR conflicts in the production RPKI. For the first scenario, the length of an IP prefix is an important factor affecting the query time of our proposed algorithm, and we can know the worst-case query times to be 32 for an IPv4 prefix and 128 for an IPv6 prefix. We randomly selected 30 IP prefixes for each prefix length to perform query operation. To investigate the effect of prefix length on query time, Fig. 4 depicts the required time for detecting INR conflicts in the first scenario for our proposed algorithm against prefix length horizon from 1 to 32 for IPv4 prefixes and 1 to 128 for IPv6 prefixes respectively. As we can see from this figure, as the length of the prefix increases, the required time for detecting INR conflicts also increases. The query time of IPv6 prefixes generally takes longer than that of IPv4 prefixes, but their difference is in the microsecond range.

Since our proposed algorithm needs to build a prefix allocation tree before detecting INR conflicts, the total query time of the study group includes the build time of the prefix tree, which is compared to the query time of the control group in two scenarios respectively. Fig. 5 depicts the total required time against different query times in the first scenario for the control and study group. In this figure, we can observe that when the number of queries is small, the proposed algorithm is not advantageous because of the construction of prefix tree . As the number of query operations increases, the query time of the control group increases sharply, while that of the proposed algorithm increases slowly. When the number

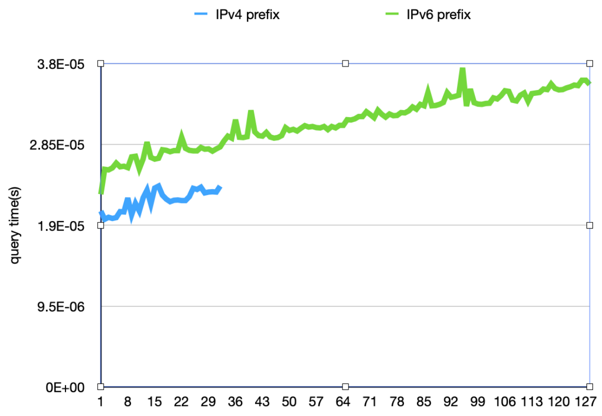

Fig. 4. Query time for a specific IP prefix with different length for the horizontal algorithm

of queries reaches 106, the total query time of the proposed algorithm is less than that of the control group, therefore, we can conclude that when INR conflict detection operations are performed in large quantities, our proposed algorithm has a great advantage and can significantly reduce the query time. In addition, the average required time for detecting INR conflicts for all IP prefixes is 467.80s (466.91s is build time and 0.89s is query time) with our proposed algorithm against 2613.24s with the control group, reducing by about 6 times.

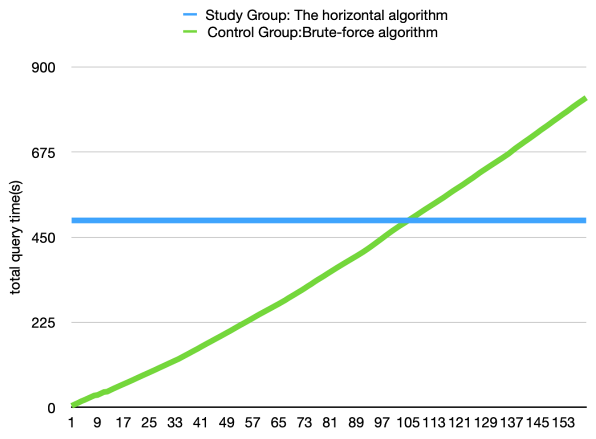

Fig. 5. Total required time against different query times in the first scenario for the study group and the control group

D. Deployment

Due to the input of the horizontal algorithm are all legal INR allocation and INR authorization information, which are expressed in RCs and ROAs that have passed the vertical algorithm, the best place to deploy the horizontal algorithms is in RP software. Since the horizontal algorithm is used as the supplement to further filter more INR conflicts, which does not affect validation results of the vertical algorithm, and can provide monitoring and early-waring mechanisms for RPKI participants, which can be used as an incentive to motivate the deployment of this algorithm, namely the RP

TABLE I
STATISTICAL ANALYSIS OF INR CONFLICTS.

Type of INR Conflicts	No. of INR Conflicts	No. of Conflicting Prefixes	RIR Distribution of INR Conflicts
B (real)	2	2	APNIC and RIPE: 1;APNIC and ARIN:1
C (real)	5	3,695	APNIC:3;RIPE:1;LACNIC:1
D (potential)	248	1,246	APNIC: 75;ARIN:32;RIPE:58;LACNIC: 71;AFRINIC: 12
E (real)	4	16	APNIC:2;RIPE:2

can improve diversity and quality of its service by providing horizontal INR conflict-detection service, hence we believe the horizontal algorithm is practical to be deployed. Our proposed algorithm also supports incremental deployment, and the RP that supports this algorithm can filter more illegal INR conflicts for ASes within its service scope. In particular, when RPs used by tier-1 ISPs support this algorithm, the illegal INR allocation and authorization relationships on the Internet will be greatly reduced.

In addition, CA operators, INR holders and network operators have some flexibility in using the INR conflict detection service, for example, they can deploy their own RP (private RP) or entrust a third-party RP (public RP) to periodically detect conflicts of INRs they hold. In reality, RPKI participants generally only care about whether there exist conflicts with INRs they hold, and the third-party RP can also provides the service to detect all INR conflicts in the RPKI system. In the first case, the number of queries is relatively small, which is general equal to the number of IP prefixes the INR holder owns, therefore, a hybrid approach can be used to support small queries using brute-force algorithm while in the background to build up the prefix tree to respond more quickly to subsequent large number of queries.

E. Statistics and Analysis of INR Conflicts

In this subsection, we test the horizontal algorithm on the production RPKI and identify real-life CA malfunctions. We explored the existence of INR conflicts (B, C, and E) and the potential existence of INR conflict D in the current RPKI deployment environment. We measured and analyzed INR conflicts from several aspects, including type, quantity, RIR distribution, and false-positive error rate. More detailed information is given in Table I, which shows types, the number and RIR distribution of INR conflicts, and the number of involved conflicting prefixes. The main reasons for the non-existence of INR conflict F we speculate are partial RPKI deployment and the recent proposal of the AS0 ROA mechanism.

The INR conflicts caused by IP address transfers are considered as legal, and our proposed algorithm cannot distinguish between them, which results in false positives. Therefore, we need to determine whether the INR conflict is an IP address transfer out-of-band and it is necessary to find the contact information of holders who owns conflicting INRs. However, objects in the production RPKI have no labels to indicate whether they are involved in INR conflicts. In addition, RPKI does not require INR holders to include identity information in their RCs, and none of the CAs have issued ghostbusters records [20] that used for retrieving the contact information of INR holders. Thus, we obtained the contact information of INR holders in the third-party database, e.g. WHOIS, and then we can contact these entities by email to ask them about their involvement in INR conflicts. However, the accuracy and effectiveness of the information in the WHOIS database cannot be guaranteed, which making correctness of our proposed algorithm hard to evaluate. It can be seen that the false-positive rate of our proposed algorithm depends on the proportion of IP address transfers in all revealed INR conflicts. In addition, the accuracy of the false positive rate depends on the accuracy of the information in the third-party database. Thus, the issuance of ghostbusters objects as soon as possible and the correctness of their contents are of great significance to the accuracy of our proposed algorithm and the early-warning service of other monitoring or detection mechanisms

To measure false positives, we have sent emails to all entities involved in INR conflicts, but only RIPE NCC replied and offered us a reasonable explanation. The conflicting IP prefix (137.74.0.0/16) had been transferred from APNIC to RIPE NCC in 2016, but APNIC did not remove it from its RPKI production CA, which resulted in this cross-RIR INR reallocation. In the above case, although IP address transfer is legal, it is unreasonable for APNIC not to have re-issued its RC four years after the above IP address had been transferred. Further, this leaves the legitimate INR holder of the conflicting IP prefix(es) exposed to potential security threats. INR conflicts detected by the horizontal algorithm can serve as an early warning to the holders of conflicting INRs, and the manner of handling them is the decision of the INR holders themselves.

V. CONCLUSION AND FUTURE WORK

In this paper, we first defined and formalized some relevant concepts and rules. Then, we combed and summarized possible CA malfunctions in INR allocation and authorization and developed the INR conflict model. Based on the above work, we proposed a horizontal INR conflict-detection algorithm to reveal INR conflicts in our model and measured its build time and query time. Finally, a trace collected from an RPKI repository was used to reveal real-life INR conflicts, and we analyzed them from several aspects. The false positives also evaluated by sending out-of-band emails to the owners of conflicting IP prefixes, but owing to the characteristics of RPKI system and the limitations of third-party information services, we cannot accurately evaluate the accuracy of the proposed algorithm. Future studies can focus on how to improve the accuracy to reduce the false-positive rate.

References

[1] M. Lepinski, S. Kent, An infrastructure to support secure internet routing. RFC 6480, 2012.

[2] G. Huston, G. Michaelson, R. Loomans. A profile for x.509 pkix resource certificates. RFC 6487, 2008.

[3] G. Huston, G. Michaelson, C. Martinez, T. Bruijnzeels, A. Newton, D. Shaw. Resource public key infrastructure (rpki) validation reconsidered. RFC 8360, 2018.

[4] Zhiwei Yan, Guanggang Geng, Hidenori Nakazato, Yong-Jin Park. Secure and scalable deployment of resource public key infrastructure (rpki). J. Internet Serv. Inf. Secur., 8(1):31-45, 2018.

[5] Y. Gilad, A. Cohen, A. Herzberg, M. Schapira, H. Shulman. Are we there yet? on rpki's deployment and security. In NDSS, 2017.

[6] Taejoong Chung, E. Aben, T. Bruijnzeels, B. Chandrasekaran, D. Choffnes, D. Levin, et al. Rpki is coming of age: A longitudinal study of rpki deployment and invalid route origins. In Proc. of the Internet Measurement Conference, 2019.

[7] Xiaowei Liu, Zhiwei Yan, Guanggang Geng, Xiaodong Li. Resource Allocation Risks by CAs in RPKI and Feasible Solutions. Computer Systems&Applications., 25(8):16-22, 2016.

[8] A. Newton, C. Martinez-Cagnazzo, D. Shaw, T. Bruijnzeels, B. Ellacott. Rpki multiple "all resources" trust anchors applicability statement. draft-rir-rpki-allres-ta-app-statement-02, 2017.

[9] G. Michaelson, G. Huston, T. Harrison, T. Bruijnzeels, M. Hoffmann. A profile for Resource Tagged Attestations (RTAs). draft-michaelson-rpki-rta-01, 2019.

[10] Y. Gilad, O. Sagga, S. Goldberg. Maxlength considered harmful to the rpki. In Proc. of the 13th International Conference on emerging Networking EXperiments and Technologies, 2017.

[11] G. Huston, G. Michaelson.Validation of route origination using the resource certificate public key infrastructure (pki) and route origin authorizations (roas). RFC 6483, 2012.

[12] Policy prop-132 (AS0 for unallocated space) deployed in service. https://blog.apnic.net/2020/09/02/policy-prop-132-as0-for-unallocated-space-deployed-in-service/.

[13] SLURM file for Unallocated and Unassigned RIPE NCC Address Space. https://www.ripe.net/participate/policies/proposals/2019-08

[14] ASO Support in AFRINIC RPKI. https://www.afrinic.net/blog/457-aso-support-in-afrinic-rpki.

[15] R. Bush. Responsible grandparenting in the rpki. draft-ymbk-rpki-grandparenting-04, 2014

[16] Caida as relationships dataset. http://www.caida.org/data/as- relationships/.

[17] Rpstir2. https://github.com/bgpsecurity/rpstir2.

[18] Routeviews. http://www.routeviews.org/routeviews/.

[19] Ripe routing information service(ris). https://www.ripe.net/analyse/internet- measurements/ routing-information- service- ris.

[20] R. Bush. The Resource Public Key Infrastructure (RPKI) Ghostbusters Record. RFC 6493, 2012.