

资源公钥基础设施数据同步的改进方法研究

冷峰^{1,2,3}, 赵琦², 延志伟², 曾宇^{1,2}

(1. 中国科学院计算机网络信息中心, 北京 100190;

2. 中国互联网络信息中心, 北京 100190;

3. 中国科学院大学, 北京 100049)

摘要: 资源公钥基础设施(RPKI)依赖方需定期从资料库系统同步数据进行信息验证。为了完成数据同步, 当前主流的方式是采用 Rsync 和 RRDP 两种技术, 但各自存在着相关问题。针对上述问题, 通过分析研究依赖方从资料库同步数据的方式, 建立了数学模型, 并根据两种技术各自面临的相关问题, 提出了一种改进的 RPKI 数据同步方法, 分析了传统数据同步手段与改进方法各自的优缺点以及适应的场景, 为优化 RPKI 的部署应用提供了参考。

关键词: 资源公钥基础设施; 路由决策; 数据同步; 数学模型

中图分类号: TN915.08

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2021064

Research on improved scheme of resource public key infrastructure data synchronization

LENG Feng^{1,2,3}, ZHAO Qi², YAN Zhiwei², ZENG Yu^{1,2}

1. Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China

2. China Internet Network Information Center, Beijing 100190, China

3. University of Chinese Academy of Sciences, Beijing 100049, China

Abstract: RPKI Relying party needs to synchronizing data from repository periodically to verify information. In general, Rsync and RRDP are the two common means to complete data synchronization, however, each of them has related problems. In order to solve these problems, through analyzing the way for synchronizing data from repository by the relying party, a mathematical model was established. Furthermore, based on the current problems faced by the two synchronization means, an improved RPKI data synchronization scheme was proposed. The advantages and disadvantages of the improved scheme were analyzed in detail, as well as the applicable scenarios. The improved scheme could provide a reference for optimizing the deployment and application of RPKI.

Keywords: resource public key infrastructure, routing decision, data synchronization, mathematical model

收稿日期: 2020-12-22; 修回日期: 2021-03-23

通信作者: 曾宇, zengyu@cnnic.cn

基金项目: 北京市科技新星计划项目(Z191100001119113)

Foundation Item: Beijing Nova Program of Science and Technology (Z191100001119113)

论文引用格式: 冷峰, 赵琦, 延志伟, 等. 资源公钥基础设施数据同步的改进方法研究[J]. 网络与信息安全学报, 2021, 7(3): 123-133.

LENG F, ZHAO Q, YAN Z W, et al. Research on improved scheme of resource public key infrastructure data synchronization[J]. Chinese Journal of Network and Information Security, 2021, 7(3): 123-133.

1 引言

互联网是由众多计算机网络相互连接组成的开放性网络, 如何进行网间寻址是解决互联网互联互通的关键问题, 其中边界网关协议 (BGP, border gateway protocol) 是互联网网间寻址的事实性标准^[1]。

由于在设计之初并未充分考虑安全问题, BGP 存在较大的安全缺陷^[2], 所导致的安全事件时有发生, 典型事件包括: 2018 年亚马逊权威域名服务器被劫持事件, 2019 年 Verizon 的错误操作导致 Facebook、AWS 等公司遭遇连锁性灾难故障以及 2020 年俄罗斯电信运营商路由劫持事件等^[3]。以 2020 年的安全事件为例, 俄罗斯运营商 Rostelecom 疑似发生路由劫持事件, 包括谷歌、亚马逊、脸书等共计 200 家 CDN 供应商流量被牵引至俄罗斯。整个事件持续约 1 h, 影响路由条目约 8 000 条。

为增强域间路由系统安全, 多项研究针对 BGP 层面的缺陷提出解决方案^[4-5], 其中, 资源公钥基础设施 (RPKI, resource public key infrastructure) 是较为成熟且已付诸实践的安全技术之一, 是目前最有可能大规模推广应用的增强路由安全的技术解决方案^[6]。

RPKI 通过构建公钥证书体系完成对互联网号码资源 (INR, internet number resource) 包括 IP 前缀和 AS 号的所有权和使用权的认证, 并以此“认证信息”指导 BGP 路由器的路由决策, 帮助其检验 BGP 报文中路由源信息的真实性^[7]。RPKI 的体系结构可主要被划分为 (CA, certification authority)、资料库 (repository) 和依赖方 (RP, relying party) 3 个基本功能模块。CA 负责签发证书标识 INR 的分配关系, 以及签发路由源授权 (ROA, route origin authorization), 授权某个 AS 针对自己的 IP 前缀发起路由起源通告等; Repository 用于存储 CA 发布的各种包含 INR 分配信息和授权信息的证书, 以及 ROA、证书吊销列表 (CRL, certificate revocation list) 等数字签名对象, 提供给全球 RP 进行同步和验证; RP 负责从 Repository 同步 RPKI 的数字签名对象并进行验证, 最终把验证

结果提供给 BGP 路由器, 用于指导 BGP 进行可信路由决策^[8]。

RP 执行签名对象的验证, 因此需要获得所有 Repository 公布的签名对象数据。一旦 RPKI 在世界范围内大规模部署, 相应的数据传输规模将迅速增大, 对互联网带宽以及 Repository 的处理能力等造成较大压力。针对 RPKI 数据同步的优化对于保障 RPKI 在大规模部署实施后维护互联网的稳定运行具有重要意义。

2 RPKI 数据同步方法概述

目前主流的 RPKI 数据同步方式是 Rsync 以及 RRDP (RPKI repository delta protocol)。针对 RPKI 数据同步的研究数量偏少, 是由于 RPKI 部署率仍然较低, 数据同步未成为 RPKI 发展过程中的主要瓶颈。Kristoff 等^[9]针对 RP 的部署情况进行全面评估, 研究了生产环境中主流 RP 通过 Rsync 以及 RRDP 等协议进行数据同步的频率、RP 在数据同步容错性方面的表现等, 但并未针对 RP 数据同步效率方面提出解决方案。

2.1 Rsync 数据同步方式存在的问题

RFC 6480 中要求, 所有 RPKI 发布点都需要支持 Rsync, 为所有的 RP 获取签名对象提供服务^[10]。Rsync 已经在业界广泛应用, 因此在 RPKI 部署初期 Rsync 可以较好地用于数据同步, 无须另行设计一套数据同步方案。

随着 RPKI 部署工作在全球范围内的逐步开展, Rsync 显露出一些有待解决的问题, 主要表现在: Rsync 的设计目标是在服务器和客户端之间传递有限规模的数据。由于服务器端需要为每一个连接消耗大量的 CPU 及内存资源, 一旦 RP 数量急剧增加, 位于中心的服务器端将承受巨大的计算压力, 同时使 Repository 容易遭受拒绝服务攻击 (DoS, denial-of-service)。

此外, Rsync 数据同步的频率由各个 RP 运行者决定, 当前 RPKI 缺少对 Rsync 机制的标准化方案。RP^[11]通常希望能更快地与 Repository 保持同步, 这是提高 RP 缓存数据及时性的行之有效的手段。但盲目提高同步频率的同时必然带来无效同步请求的提升, 造成 Repository 系统资源的无谓消耗。此现象导致的后果与数字

证书更新频率以及 RP 数量成正比,各种因素结合造成 Repository 消耗的资源远大于合理值。同时, Rsync 数据同步的频率由各个 RP 运行者决定,根据经验分析,存在较大概率造成部分时间段 Repository 同步压力过大,其他时间段 Repository 又过于空闲,即无法合理分配 Repository 资源。

2.2 RRDP 数据同步方式存在的问题

基于 Rsync 存在的问题, IETF 针对 RPKI 的数据同步机制提出改进方案,利用增量同步 Delta 协议替代 Rsync,称作 RRDP^[12]。RRDP 利用 Update Notification (更新通知)、Snapshot (快照)以及 Delta Files (增量文件)进行数据同步,并使用 HTTPS 协议进行数据传输,便于使用如内容分发网络(CDN, content delivery network)等机制进行负载分担,降低 Repository 的传输压力。

RRDP 的更新通知文件包含唯一的会话 ID 和序列号, RP 可通过这两项的值判断是否与 Repository 处于完全同步状态。如果未完全同步, Update Notification 可以用来定位 Snapshot 及 Delta Files 的位置, RP 可进一步完成同步工作(快照文件包含了当前 RPKI 资料库中所有的对象)。RP 为获取新的数据,会定期向 Repository 发出数据同步请求。为保证 RP 能够获取到所有更新信息, Repository 需将旧版本的 Delta Files 保存较长一段时间。随着 RPKI CA 系统不断签发、更新或撤销对象,由此产生的海量的增量文件将对 Repository 的缓存基础设施提出巨大挑战^[13]。

由此可见, Rsync 对服务器的计算能力有较大需求,而 RRDP 将这种计算需求转化为较大的存储需求。两者在解决 RPKI 数据同步问题时,仍存在一定的改进空间。如何对两种协议进行融合,充分利用各自优势提出更佳的数据同步方式,是促进 RPKI 部署应用的一种有效手段。

2.3 已开展的相关工作

针对 RPKI 数据同步的问题,业界已逐步开展相关工作。司昊林等^[14]研究了增量同步协议的工作逻辑,并与 Rsync 进行全面对比,证明了 Delta

协议具备较高的安全性和稳定性。Kristoff 等^[9]针对 RP 部署情况进行测量,其中一项主要内容是统计了 RP 的刷新频率,结果发现 2019 年 Rsync 仍然是 RP 主要采用的询问方式。Louis 等^[15]分析了当前五大 RIR 机构对 RPKI 的部署状态,评估了各 RIR 对 RRDP 的支持程度,并提出了一种新的可实现 RPKI 核心功能的开源软件,可支持 RRDP/Rsync 两种同步方式。

3 RPKI 数据同步模型

为了直观理解 RPKI 数据同步的压力,本节针对 RPKI 数据同步建立数学模型,用于与后续提出优化的 RPKI 数据同步方式进行对比分析。

RPKI 完全部署实施后,所有 RPKI 相关对象的总体大小可表达如下。

$$O^{\text{Total}} = \left\{ \bigcup_{i=1}^5 \left(\overline{\text{ca}}_{\text{ICANN}}^{\text{RIR}}, \overline{\text{ca}}_{i+1}^{\text{RIR}}, \overline{\text{ca}}_{i+2}^{\text{RIR}}, \overline{\text{ca}}_{i+3}^{\text{RIR}}, \overline{\text{ca}}_{i+4}^{\text{RIR}} \right), \right. \\ \left. \overline{\text{InfraObjs}}, \overline{\text{AllocObjs}}, \overline{\text{RoutingObjs}}, \overline{\text{StructObjs}} \right\}$$

其中, $\overline{\text{ca}}_{\text{ICANN}}^{\text{RIR}}$ 表示 RIR 内部为 IANA 分配资源认证设立的 CA 所产生的对象大小(CA 证书、CRL、GhostBuster 等); $\overline{\text{InfraObjs}}$ 表示各个组织的 CA 所产生的对象大小; $\overline{\text{AllocObjs}}$ 表示资源分配所产生的对象大小; $\overline{\text{RoutingObjs}}$ 表示与证明地址宣告相关的对象大小; $\overline{\text{StructObjs}}$ 表示 RPKI 资源列表的大小。由此可见, RPKI 运作涉及众多数据“对象”的同步。为了优化 RPKI 同步机制,本文融合 Rsync 节省存储空间的优势,以及 RRDP 降低计算成本的优势,从主动通知和文件监测等角度提出方案,设计适合 RPKI 大规模部署的数据同步方法。

3.1 以 Rsync 方式进行 RPKI 数据同步的主要步骤

以采用 Rsync 为手段的数据同步为例,根据 Rsync 的同步原理, Rsync 客户端(即 RP)通过对比文件变化的方式保持与服务端的同步。假设有两台设备,其中 α 为服务端,控制文件 A; β 为客户端,控制文件 B,则 Rsync 算法的处理过程可归纳为以下几个步骤^[16]。

(1) β 将文件 B 分割成一组不重叠的固定大

小的数据块。

(2) β 对每一个分割好的数据块执行两种校验：一种是 32 bit 滚动弱校验；另一种是 128 bit MD4 强校验。

(3) β 将这些校验结果发给 α 。

(4) α 通过搜索文件 A 的所有同样固定大小的数据块，寻找与文件 B 的某一块有相同弱校验码和强校验码的数据块。

(5) α 发给 β 一串指令来生成文件 A 在 β 上的备份（以上指令或者对文件 B 的某一个数据块无须重传的证明），或者一个数据块，此数据块未和文件 B 的任何一个数据块匹配。

由以上过程可知，Rsync 的技术特性更适合于数量较少的小文件同步，在客户端与服务端之间存在相似的文件，对比后仅传输差异部分，这有助于节省客户端及服务端之间的带宽消耗。但与此同时，Rsync 对数量大、体积大的数据进行同步的效果并不理想，Rsync 在处理大量小文件进行数据同步的时间过长，时而会出现进程突然停止等异常现象。同时，Rsync 在同步大文件时容易无故中断，针对大文件同步场景的适用性和稳定性表现不尽理想。

3.2 传统 RPKI 数据同步在一个时间周期内的无效请求次数

为了进一步研究 RP 数据同步效率与各资源消耗之间的关系，以下描述 RP 采用 Rsync 机制的资源消耗状况。

假设对于单个 CA，共有 N 个 RP 将前往该 CA 进行数据同步。设 CA 产生的签名数据变化起始时间与各 RP 请求 Rsync 数据同步的起始时间相同，若 CA 产生签名数据变化的时间周期为 t^{CA} ，各个 RP 请求的时间间隔为 t_i^{RP} ， $1 \leq i \leq N$ ，则对于某个 CA 的一个数据更新周期中，第 i 个 RP 产生的无效请求次数可表示为

$$\text{times}_i^{RP} = \sum_{i=1}^N \text{DIV}(t^{CA}, t_i^{RP})$$

其中，DIV 表示整除运算，则在一个时间周期 t^{CA} 内，所有 RP 的无效请求次数可以表示为

$$\text{times}_{ALL}^{RP} = \sum_{i=1}^N \text{times}_i^{RP}$$

由上式可以看出， t_i^{RP} 与 t^{CA} 的差距越大（即 RP 的更新频率较 CA 的数据更新频率相差越大）， N 的值越大（即 RP 的数量越多），则在某个 CA 的一个数据更新周期中所产生的无效请求越多。同时其中隐含的是签名对象大小加剧了无效请求所产生的负面影响。签名对象数量越多、体积越大，则每次无效请求所涉及的文件对比操作所占有的系统资源越多，对 Repository 的无谓消耗越大。

以上的情况在 RPKI 实施过程中 CA 数据更新频率较低时更为严重。同时，一旦 RP 数量急剧增多，将进一步增加无效 Rsync 同步查询的次数，导致负面影响加剧。

3.3 传统 RPKI 数据同步的资源消耗的数学模型

对于一次典型的数据同步，其主要的系统资源消耗在于 RP 数据同步请求的接收，RP 数据块、Repository 数据块的对比以及对比结果回传等 3 个动作。

以 Rsync 同步方式为例，在某一 RP 发起数据同步请求后，Repository 的主要资源消耗可分为三个步骤：① 接收 RP 的数据同步请求；② 对比数据块异同；③ 返回无须重传的证明，或者对应数据块。

以下描述在一个时间周期 t^{CA} 内 Repository 资源消耗的基本情况。

假设各个 RP 在 t^0 时刻已与 Repository 完成数据同步，在第一个 t^{CA} 内，各 RP 发来同步请求后，Repository 经对比返回给 RP 无须重传的证明，则后续不会产生无效消耗。因此，对于资源消耗的基本情况应从第二个时间周期开始。

对于第 i 个 RP，在第一个时间周期 t^{CA} 内所产生的请求次数（即上文计算的无效请求次数）为 times_i^{RP} ，则在第二个时间周期内第 i 个 RP 产生第一次同步请求的时间为

$$t_i^1 = (\text{times}_i^{RP} + 1) \cdot t_i^{RP}$$

则在此周期内第 i 个 RP 发起的所有同步请求次数为

$$\text{times_sync}_i^{RP} = \text{DIV}(2t^{CA} - t_i^1, t_i^{RP})$$

每次同步请求发生的时间为

$$t_sync_i^{RP} = t_i^{RP} \cdot \text{quantity}, \quad \text{quantity} \in [\text{times}_i^{RP} +$$

$1, \text{times}_{\text{sync}_i}^{\text{RP}}]$

此外, 3 个典型步骤对系统产生的压力各不相同。步骤①和步骤③主要涉及网络传输, 对于 CPU、I/O 等资源消耗相对较小。而步骤②主要涉及数据块对比, 相对来说, CPU、I/O 资源的消耗明显高于步骤①和步骤③。为突出主要矛盾, 以下假设各 RP 数据同步请求包大小统一为 $\text{size}_{\text{rsync}}$, 各 RP 同步数据块大小统一为 $\text{size}_{\text{reply}}$, 其处理时间可被表示为

$$t_{\text{rsync}} \text{ or } t_{\text{reply}} = \text{Bandwidth}_{\text{Repo}} / (\text{Nsize}_{\text{rsync}} \text{ or } \text{Nsize}_{\text{reply}})$$

设数据块异同对比时间为 t_{compare} 。

对于任意 RP, 其一次常规的数据同步经历的总时长可以表示为

$$\text{time}_{\text{onesync}} = t_{\text{rsync}} + t_{\text{compare}} + t_{\text{reply}}$$

则一轮数据同步的截止时间可以表示为

$$\text{time}^{\text{finish}} = t_{\text{sync}_N}^{\text{RP}} + \text{time}_{\text{onesync}}$$

由于本阶段主要评估 Repository 的压力, 上述表示法忽略了网络传输时间等其他因素。其中, 对于一个周期 t^{CA} 内, 3 种动作首次开始与结束的时间, 可用以下式子表示。

$$t_i^1 \leq \text{time}_{\text{range}}^{\text{rsync}} \leq t_i^1 + t_{\text{rsync}}$$

$$t_i^1 + t_{\text{rsync}} \leq \text{time}_{\text{range}}^{\text{compare}} \leq t_i^1 + t_{\text{rsync}} + t_{\text{compare}}$$

$$t_i^1 + t_{\text{rsync}} + t_{\text{compare}} \leq \text{time}_{\text{range}}^{\text{reply}} \leq t_i^1 + t_{\text{rsync}} + t_{\text{compare}} + t_{\text{reply}}$$

因此, 在一个周期内, 有效同步的结束时间为

$$\text{endtime}_{\text{lastround}}^{\text{rsync}} = t^{\text{CA}} + \max \text{time}_{\text{range}}^{\text{rsync}}$$

结合任意 RP 每次发起的数据同步请求的时间 $t_{\text{sync}_i}^{\text{RP}}$, 若计算在某一时刻 t , Repository 所承载的所有数据同步的任务量 consumption_t , 需知道在此时 Repository 针对各个 RP 所进行的相应操作, 即判断在 t 时刻, 针对 RP_i , Repository 正在进行的操作是 $\{\text{act}_{\text{rsync}}, \text{act}_{\text{compare}}, \text{act}_{\text{reply}}, \text{act}_{\text{nothing}}\}$ 中的哪一类。判断的依据是

$$\text{act}_t^i = \begin{cases} \text{act}_{\text{rsync}}^{\text{rsync}}, t \in \text{time}_{\text{range}}^{\text{rsync}} \\ \text{act}_{\text{compare}}^{\text{compare}}, t \in \text{time}_{\text{range}}^{\text{compare}} \\ \text{act}_{\text{reply}}^{\text{reply}}, t \in \text{time}_{\text{range}}^{\text{reply}} \\ \text{act}_{\text{nothing}}^{\text{nothing}}, t \in \text{其他} \end{cases}$$

则可以将 t 时刻 Repository 承载的所有数据同步的任务量 consumption_t 表示为

$$\text{consumption}_t = \sum_{i=1}^N \text{act}_t^i$$

以上分析的是在一个周期 t^{CA} 内有效动作的任务量, 即首次同步产生的 3 个动作的描述。此周期内只有一次 RPKI 数据更新, 产生的其他动作可被认为是无效动作, 暂不分析由其产生的任务量。

4 RPKI 数据同步的改进方案

针对传统 RPKI 数据同步存在的问题, 本文提出了一种 RPKI 数据同步的改进方案, 改进点可概括为以下几方面。

(1) 建立主动通知机制, 利用 Repository 建立 RP 资料库, 由 Repository 主动发起数据同步通知。

(2) 提出慢启动数据传输机制, 在 Repository 主动发起数据同步通知时, 按照批次向 RP 发出同步通知。

(3) 改进方案的有效性和效能可通过教学模型及仿真实验加以验证。

(4) 设立增量文件监测系统, 及时删除访问频率低的增量文件。

(5) 根据以上机制存在的安全风险增加 RP 安全认证机制。

其中, (1)、(2) 方面用于减少 RP 对 Repository 的无效同步请求, 降低 Repository 的无效负载, 第 (4) 方面用于缓解 Repository 存储空间压力, 第 (5) 方面用于增强系统安全。

4.1 建立主动通知机制

在初始阶段, Repository 并不知道 RP 的具体信息, 因此 RP 一开始需自发发起对 Repository 的数据同步请求。RP 在前往某一 Repository 获得数据的同时, Repository 需记录 RP 的 IP 地址, 并将其添加至 RP 列表。Repository 在产生新的更新通知时, 将更新通知文件的 URI 主动发送给

列表中的 RP。在 RPKI 广泛部署、RP 数量众多的情况下,可采用如 CDN 等降低发送主动通知给 RPKI 资料库带来的压力。当产生数据更新时,RPKI 资料库将触发 CDN 机制为列表中的 RP 发送主动通知。

4.2 提出慢启动数据传输机制

在 RRDP 中,RP 端的数据同步依靠 RP 定期前往 Repository 拉取实现,Repository 仅能被动提供相关数据,缺乏统一协调机制,易出现由于各 RP 数据同步时间重合而引发的 Repository 瞬时压力陡增,或 RP 查询频率过高而导致 Repository 产生无谓的资源浪费等情形。第 3 节利用数学模型讨论了 RP 产生的无效请求次数。但 RPKI 数据同步模型可被归类为客户端/服务器模式(C/S 模式)。位于中心的 Repository 作为服务器端,要同时响应来自各个区域的 RP 的同步请求。考虑到 RP 的数据同步时间将因 Repository 负载的动态变化而受到相应影响,在有数据更新时统一向所有 RP 发出更新通知并不是最优解决方案。为了避免同步操作过于集中,优化数据同步表现,本文提出慢启动机制,在 Repository 主动发起数据同步通知时,按照批次向 RP 发出同步通知。

4.3 RPKI 数据同步改进方案的数学模型

根据数据同步处理机制原理可知,在第一个签名对象更新周期 t^{CA} 内将不主动产生数据同步通知,数据同步通知将从 $2t^{CA}$ 开始分批次发出。Repository 运营者应根据其系统实际处理能力以及 RP 数量确定每次更新通知的 RP 数量 N^{round} 。

在 Repository 的一次更新通知发生后,其经过的时间可以由通知发出的时间、通知传输的时间、RP 发来数据传输的时间以及数据接收时间组成。因为本阶段主要关注 Repository 的处理压力,以下研究将忽略通知发出的时间、通知传输以及 RP 发来的数据传输的时间,仅计算数据接收时间 $newt_{receive}$ 。Repository 在一轮同步周期中向 RP 发出更新通知的处理时间可以表示为

$$newt_{rsyc} = \text{Bandwidth}_{\text{Repo}} / (N^{round} \cdot \text{newsize}_{rsyc})$$

对于返回数据块操作的持续时间可用同样的公式表示。

本文方法以分批次的形式向 RP 发起数据同步通知,因此在第一个签名对象更新周期 t^{CA} 内没有通知发出。由于向第 1 至 N^{round} 的 RP 一次发出同步通知的时间为 t^{CA} ,可以将向所有 RP 发出同步通知的时间表示为

$$newt_i^1 = \begin{cases} t^{CA}, 1 \leq i \leq N^{round} \\ t^{CA} + newt_{rsyc}, N^{round} + 1 \leq i \leq 2N^{round} \\ t^{CA} + 2newt_{rsyc}, 2N^{round} + 1 \leq i \leq 3N^{round} \\ \vdots \\ t^{CA} + Pnewt_{rsyc}, PN^{round} + 1 \leq i \leq N \end{cases}$$

其中, P 为 N 与 N^{round} 的整除运算,即

$$P = \text{DIV}(N, N^{round})$$

基于以上式子和前文表述,可以计算完成一轮数据更新的截止时间为

$$\text{time}_{\text{new}}^{\text{finish}} = newt_N^1 + t_{\text{compare}} + t_{\text{reply}}$$

将新旧两种方法完成一轮数据更新的截止时间相比较,可以看出,当公式右侧第二、三个变量相同时,数据更新总时间主要由第一个变量决定。而影响第一个变量的主要因素包括 RP 数量、Repository 处理能力以及签名对象更新频率等。在 RP 数量较少、签名对象更新频率较低以及 Repository 处理能力较强的场景,原数据同步手段的处理总时长较短,反之则新的数据同步方式更具优势。并且,在一个签名对象更新周期内一旦完成数据同步,新方案不会产生多余的无效更新通知,而原有方案则仍会源源不断地定期产生无效更新请求。

同样,可以计算在任意时刻 t ,新的同步方案中 Repository 所承载的所有数据同步的任务量 $\text{consumption}_t^{\text{new}}$,整个过程仍可以被分为 3 个阶段:① 发送数据同步更新通知,并接收 RP 发来的同步请求;② 对比数据块异同;③ 返回数据块。

对于第一阶段产生的动作 $\text{act}_t^{\text{rsyc}}$,其开始时间是签名对象更新的时间,即

$$\text{starttime}_{\text{firstround}}^{\text{rsyc}} = t^1, t^1 = t^{CA}$$

其结束时间是最后一批向 RP 发送数据更新通知的结束时间,即

$$\text{endtime}_{\text{lastround}}^{\text{rsync}} = t^1 + (P+1)\text{newt}_{\text{rsync}}$$

则

$$\text{starttime}_{\text{firstround}}^{\text{rsync}} \leq \text{newtime}_{\text{range}}^{\text{rsync}} \leq \text{endtime}_{\text{lastround}}^{\text{rsync}}$$

根据上述公式, 可以计算针对第二阶段, 即对比数据块异同动作的起始时间, 按照每轮动作作为一个单位, 表示为

$$\text{new_comparat}_i^1 = \begin{cases} t^{\text{CA}} + \text{newt}_{\text{rsync}}, 1 \leq i \leq N^{\text{round}} \\ t^{\text{CA}} + 2\text{newt}_{\text{rsync}}, N^{\text{round}} + 1 \leq i \leq 2N^{\text{round}} \\ t^{\text{CA}} + 3\text{newt}_{\text{rsync}}, 2N^{\text{round}} + 1 \leq i \leq 3N^{\text{round}} \\ \vdots \\ t^{\text{CA}} + (P+1)\text{newt}_{\text{rsync}}, PN^{\text{round}} + 1 \leq i \leq N \end{cases}$$

则其结束时间为

$$\text{endtime}_{\text{everyround}}^{\text{compare}} = \text{new_comparat}_i^1 + t_{\text{compare}}$$

即

$$\text{new_comparat}_N^1 \leq \text{newtime}_{\text{range}}^{\text{compare}} \leq \text{endtime}_{\text{everyround}}^{\text{compare}}$$

针对第三阶段, 即返回数据块动作的开始结束时间的推演过程可参考第二阶段。

$$\text{new_reply}_i^1 \leq \text{newtime}_{\text{range}}^{\text{reply}} \leq \text{endtime}_{\text{everyround}}^{\text{reply}}$$

则判断 t 时刻 Repository 正在执行的动作, 可以由以下式子表示。

$$\text{newact}_i^t = \begin{cases} \text{act}_i^{\text{rsync}}, t \in \text{newtime}_{\text{range}}^{\text{rsync}} \\ \text{act}_i^{\text{compare}}, t \in \text{newtime}_{\text{range}}^{\text{compare}} \\ \text{act}_i^{\text{reply}}, t \in \text{newtime}_{\text{range}}^{\text{reply}} \\ \text{act}_i^{\text{nothing}}, t \in \text{其他} \end{cases}$$

则可以将 t 时刻 Repository 所承载的所有数据同步的任务量 new consumption_t 表示为

$$\text{newconsumption}_t = \sum_{i=1}^N \text{newact}_i^t$$

注: 以上同样仅考虑有效的数据同步。

若比较新的数据同步方式以及传统数据同步方式在 t 时刻实际承载的任务量大小, 需对比 consumption_t 和 newconsumption_t 两个变量。

对于传统数据同步方式, 当 t 属于 $\{\text{time}_{\text{range}}^{\text{rsync}}, \text{time}_{\text{range}}^{\text{compare}}, \text{time}_{\text{range}}^{\text{reply}}, \text{其他}\}$ 任意一组区间时, 最多同时执行的动作数量为 N 个 (未考虑无效 RP 同步请求)。而对于新的数据同步方式, 当 t 属于 $\{\text{newtime}_{\text{range}}^{\text{rsync}}, \text{newtime}_{\text{range}}^{\text{compare}}, \text{newtime}_{\text{range}}^{\text{reply}}, \text{其他}\}$ 任意一组区间时, 最多同时执行的动作数量为 N^{round} 个 (以上皆未考虑无效 RP 同步请求)。可以看出, 只要 $N^{\text{round}} < N$, 新的数据同步方式中 Repository 在 t 时刻所承担的任务量 $\text{newconsumption}_t < \text{consumption}_t$ 。

从两者同步方式的原理可以直观得出以下结论: 由于新的数据同步方式将 RP 的同步请求进行分时处置, 对于 t 时刻承载的任务量明显小于原数据同步方式, 在频繁产生签名对象更新以及大量 RP 需开展数据同步请求的情形下, 可有效优化 Repository 负载。

为对比两种数据同步方式的差异, 假设条件如下: ① 共有 10 个 RP 需进行数据同步; ② 传统 RPKI 同步周期为 4 min; ③ 以 Repository 系统处理能力为基准, 其总处理表示为 100%。④ rsync、compare、reply 所占用的系统处理能力 (即系统消耗量) 各为 10%。

根据以上假设, 结合前文描述的两种数据同步方式的数据量描述, 两种数据同步方式下 Repository 系统负载情况如图 1 所示。其中, 点状绿色面积代表传统 RPKI 数据同步机制下在同

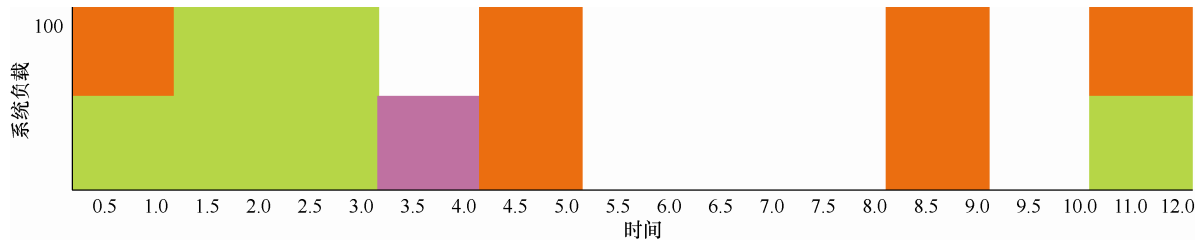


图1 RPKI 数据同步系统负载比较
Figure 1 RPKI data synchronization system load comparison

步过程中的系统消耗情况,紫色区域面积代表改进 RPKI 数据同步机制下在同步过程中的系统消耗情况,绿色区域代表两种数据同步方式同时占用的系统消耗情况。可以看出,相较于传统同步机制,改进的同步方法可以更加合理地调用系统资源,同时可有效减少在同步空闲时间对 Repository 系统的无效占用,降低系统无谓消耗。

4.4 设立增量文件监测系统

RFC 8182 中建议,当增量文件大小总和大于快照文件大小时,RP 不应继续采用增量更新的方式进行同步,以免造成不必要的文件传输^[17]。此时应启动清理机制,将早期的增量文件删除,从而保证 RP 请求增量文件的总消耗始终小于请求快照文件的消耗,从而降低系统负载。以上机制对于 RRDP 数据同步方式而言,一方面避免了增量更新情形下不必要的数据传输,另一方面提出了较小限度地减小 Repository 磁盘空间占用的清理机制。但随着 RPKI 的逐步推广,快照文件将根据 RPKI 对象数量多少以及变化频率逐步变大,相应对于增量更新文件的清理周期会逐步变长。根据本文提出的主动通知机制,常规情况下,绝大多数 RP 开展新的数据同步操作时以访问较新的增量文件为主,较早版本的增量文件被访问的概率极低,将长时间存储在 Repository 中,对资料库造成不必要的负载。为了进一步优化系统占用,本文提出增量文件监测系统。

增量文件监测系统主要通过监测一段时间内早期增量文件被查询的情况,结合磁盘空间占用量决定是否主动删除无请求或请求频率低的增量文件。假设在时间内,若增量文件 f_i ($1 < i < M$) 被查询的次数 $\text{query}_{f_i} = 0$,则表示在 Δt 内增量 f_i 都处于闲置状态。根据原理可知, f_i 以及较其更早的增量文件在未来的一段时间内被查询的可能性较低,因此可将其选择性删除,以降低系统磁盘占用及无效负载。对于监测时间周期 Δt 的设置,可以根据资料库的访问量以及本身硬件资源进行适当调整。相比原来的文件清理机制,设立增量文件监测系统可以更好地对使用频率过低的增量文件进行清理。

4.5 增加 RP 安全认证机制

增加的主动通知机制将成为数据同步的关键环节,即通知功能集中在 Repository 处。假设不增加任何限制,黑客可通过伪造 RP 的方式短时间内向 Repository 发送大量同步请求,导致 Repository 无法响应正常同步请求,造成服务中断。为缓解以上威胁,在增加主动通知机制的同时,应辅认证机制。具体做法为:首先验证 RP 端的合法身份,如利用带外的方式令 RP 端完成身份认证,成功后加入合法 RP 地址列表;其次为每一个合法 RP 派发数据同步密钥,在 Repository 主动发起数据同步通知时,双方利用密钥完成身份认证后方可进行数据传输。

需要注意的是,在现有的 RPKI 数据同步机制中,Repository 对外提供公开类别的服务,并未设定关于 RP 的身份认证机制。为了能与现有机制充分保持一致,Repository 应最大限度响应由 RP 主动发起的数据同步请求,此时的安全防护策略与现有机制下的防护策略并无明显差异。

5 仿真实验

为验证改进方案的有效性,以下搭建模拟环境后并利用仿真实验开展验证。设计思路为使用虚拟机建立 Repository 以及 RP,采用传统 RPKI 数据同步方式以及改进方案分别完成数据同步,对比两种方式下 Repository 系统的 CPU 以及 I/O 负载变化情况。为了突出主要矛盾,以下选定 Rsync 同步方式与改进方案进行性能对比。

以两台 x86 主机搭建实验环境,其中一台主机承载 Repository 功能,另一台承载 RP 功能,实验环境系统架构如图 2 所示。

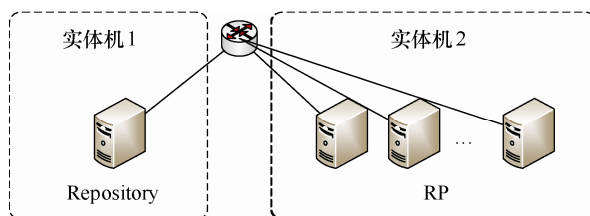


图 2 实验环境系统架构
Figure 2 Experimental environment system architecture

实验环境系统信息如表 1 所示。

为了对比两种同步方式的差异,设定以下两种

表 1 实验环境系统信息
Table 1 Experimental environment system information

同步方式	CPU	内存	磁盘	操作系统	网络
Repository	i7 7700 单核模拟	2 GB	20 GB SSD	CentOS7	1 000 Mbit/s
RP	i7 7700t 单核模拟	2 GB	20 GB SSD	CentOS7	1 000 Mbit/s

场景模拟各自同步操作。

(1) 对于 Rsync 同步方式, Repository 目录中存储同步文件大小为 200~500 MB, 更新频率 3 min; 所有 RP 同时更新, 更新频率设置为 1 min。为了同时对比加密传输时对 Repository 系统的负载情况, 设定了在启用/非启用加密数据传输时的同步场景。

(2) 对于改进的同步方式, Repository 目录中存储同步文件以及 RP 设定与上一种情况一致。设定 Repository 向所有 RP 发起同步通知, 以分组的方式发出通知信息, 每 3 个 RP 为一组。

以上情形设定可在相似基础条件下客观评测两种更新方式为 Repository 带来的压力。

经过多轮评测, 针对不同情形, 分别记录 Repository 系统压力以及传输时间。整个测试期间为三组更新, 其中每个周期内文件的大小不一。图 3 展示了改进方式下的测试情况。

如图 3 所示, 在改进方式下, 共统计了 Repository 在 3 个变更周期下的系统占用情况, 主要以 CPU 占用率为主要参考依据。从图中数据可以看出, 第一个周期于 108 s 完成同步, 第二个周期于 82 s 完成同步, 第三个周期于 73 s 完成同步, 同时, 系统资源占用较为稳定, 整体占用率小于 50%。

图 4 展示了默认的 Rsync 同步方式的测试情况。在默认的 Rsync 同步方式下, 第一个周期于 114 s 完成同步, 第二个周期于 118 s 完成同步, 第三个周期于 100 s 完成同步, 系统资源占用相对有较大波动, 但平均系统占用在 30% 以下, 部分时刻系统占用率突发至 45% 左右。可以看出, 由于 Rsync 未使用加密传输方式, 其对系统资源的消耗小于改进方式, 但 3 个周期的传输速率相对改进方式分别有 5.56%、43.90%、36.99% 的滞后。随着 RP 同步时间的逐步延长, 改进方式的同步效率优势则更为明显。

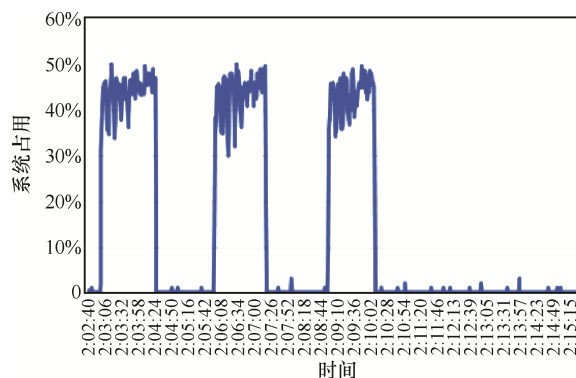


图 3 改进方式的系统占用统计
Figure 3 Improved mechanism occupancy statistics

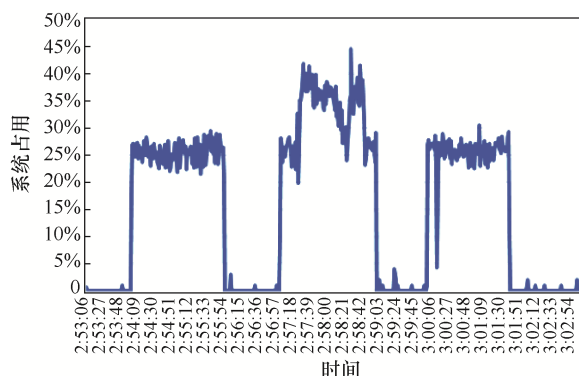


图 4 Rsync 同步的系统占用统计
Figure 4 Rsync synchronization system occupancy statistics

图 5 展示启用加密传输后 Rsync 同步的测试情况, 此举是为了侧面验证在实施安全手段后对系统性能的损耗。带有加密的 Rsync 同步方式下, 第一个周期于 153 s 完成同步, 第二个周期于 118 s 完成同步, 第三个周期于 101 s 完成同步, 系统资源占用相对稳定, 平均系统占用在 60% 以下。可以看出, 在使用加密手段后, Rsync 对系统资源的消耗进一步加大, 同时 3 个周期的传输速率相对改进方式分别有 41.67%、43.90%、38.36% 的滞后, 加密方式对于数据传输造成明显影响。

同时从图 5 中可以看出, Rsync 在周期外系统资源的小幅占用明显高于改进方式, 这是由于无效 Rsync 对比操作而形成的资源浪费, 改

进方式可以显著消除此部分系统占用,进一步优化系统效率。

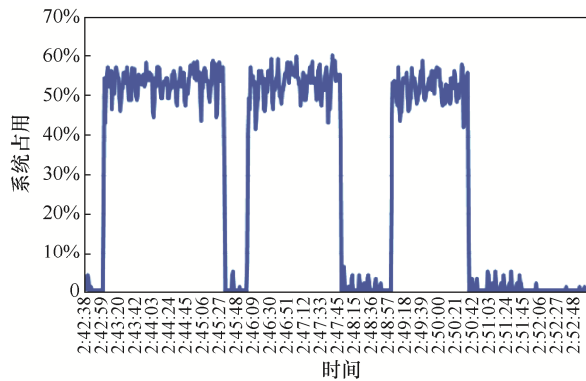


图5 带加密的 Rsync 同步的系统占用统计
Figure 5 System occupancy statistics of encrypted Rsync synchronization

实验数据统计结果如表2所示。

表2 实验数据统计结果
Table 2 Result of experimental data statistics

同步方式	周期一	周期二	周期三
改进方式	108 s	82 s	73 s
Rsync	114 s	118 s	100 s
加密的 Rsync	153 s	118 s	101 s

6 结束语

本文针对 RPKI 数据同步进行研究,分析了 Rsync 以及 RRDp 各自面临的问题,并提出针对性改进方案,同时利用数学模型对比了新旧数据同步方法的优劣,指出了各自适应的应用环境。

本文的研究存在两方面局限:(1)为了突出主要问题,研究过程中对于某些因素进行了部分简化,如未过多考虑网络传输时延的影响,针对 Repository 压力的计算也仅测算了有效传输的部分;(2)未在现行 RPKI 生产环境中进行实际验证。后续的研究工作可以从以上方面着手改进。一是可以进一步全面评估改进算法对 Repository 以及 RP 产生的影响,可结合当前 Repository 和 RP 的实际数量及负载等现状进行综合评估;二是可以在生产环境中搭建原型系统,使部分 Repository 与 RP 间利用改进方法进行数据同步并

供少量用户实际使用,验证改进方案的有效性和稳定性。若能有效解决上述问题,RPKI 数据同步改进方案可逐步在生产环境中推广应用,成为优化 RPKI 数据传输的有效手段。

参考文献:

- [1] 庞超, 李晓红. BGP4+协议研究与应用[J]. 无线电工程, 2011, 41(2): 4-6.
PANG C, LI X H. Study and application of BGP4+ protocol[J]. Radio Engineering, 2011, 41(2): 4-6.
- [2] 黎松, 诸葛建伟, 李星. BGP 安全研究[J]. 软件学报, 2013, 24(1): 121-138.
LI S, ZHUGE J W, LI X. BGP security research[J]. Journal of Software, 2013, 24(1): 121-138.
- [3] SARA B, HAFSSA, MOUAD B M. Security problems in BGP: An overview[C]//2013 National Security Days (JNS3). 2013: 1-5.
- [4] IETF. Secure BGP (S-BGP)[S].
- [5] IETF. Extensions to BGP to support secure origin BGP (soBGP)[S].
- [6] RIPE NCC. Number of certificates[S].
- [7] 贾佳, 延志伟, 耿光刚, 等. 一种改进的 BGP 路由源认证机[J]. 计算机系统应用, 2017(1): 240-245.
JIA J, YAN Z W, GENG G G. An improved BGP routing source authentication mechanism[J]. Computer Systems Application, 2017(1): 240-245.
- [8] 苏莹莹, 李丹, 叶洪琳. 资源公钥基础设施 RPKI: 现状与问题[J]. 电信科学, 2021, 37(3): 75-89.
SU Y Y, LI D, YE H L. Resource public key infrastructure RPKI: status and problems[J]. Telecommunications Science, 2021, 37(3): 75-89.
- [9] KRISTOFF J, RANDY B, CHRIS K, et al. On measuring RPKI relying parties[C]//Proceedings of the ACM Internet Measurement Conference(IMC '20). 2020: 484-491.
- [10] IETF. An infrastructure to support secure internet routing[S].
- [11] WEBERB, BRUIJNZEELST, MURAVSKIYO. RPKI repository analysis and requirements[R].
- [12] IETF. The RPKI repository delta protocol (RRDP)[S].
- [13] OSTERWEIL E, MANDERSON T. Sizing estimates for a fully deployed RPKI[R].
- [14] 司昊林, 马迪, 毛伟, 等. RPKI 增量同步 Delta 协议的形式化检测与实现[J]. 计算机系统应用, 2018, 27(11): 1-8.
SI H L, MA D, MAO W. Formal verification and implementation of RPKI incremental synchronous delta protocol[J]. Computer Systems Application, 2018, 27(11): 1-8.
- [15] LOUIS P, MARTIN J L. Cloudflare and RPKI at scale[EB].
- [16] 王宾, 刘钊远. 基于 Rsync 的远程文件同步优化模型[J]. 计算机与现代化, 2015(4): 10-13.
WANG B, LIU Z Y. Optimization model of remote file synchronization based on Rsync algorithm[J]. Computer and Modernization, 2015(4): 10-13.
- [17] PHOKEERA. Interdomain routing security: motivation and challenges of RPKI [D]. Royal Holloway, University of London, 2014.

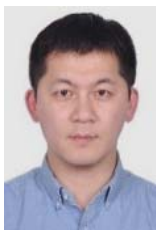
[作者简介]



冷峰（1982- ），男，山东莱阳人，中国科学院大学博士生，中国互联网络信息中心高级工程师，主要研究方向为互联网基础资源安全。



延志伟（1985- ），男，山西兴县人，博士，中国互联网络信息中心研究员，主要研究方向为互联网名址协议及下一代网络架构。



赵琦（1982- ），男，吉林长春人，中国互联网络信息中心高级工程师，主要研究方向为系统架构设计、优化和网络安全。



曾宇（1973- ），男，湖南邵阳人，博士，中国互联网络信息中心研究员，主要研究方向为计算机体系结构、网络安全、数字经济。