计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(11):1−8 [doi: 10.15888/j.cnki.csa.006562]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

# RPKI 增量同步 Delta 协议的形式化检测与实现①

司昊林[1,2,3], 马　迪[2], 毛　伟[2,3], 王　伟[2], 邵　晴[3]

[1](中国科学院 计算机网络信息中心, 北京 100190)
[2](中国科学院大学, 北京 100049)
[3](互联网域名系统北京市工程研究中心, 北京 100190)
通讯作者: 司昊林, E-mail: 1335800763@qq.com

**摘　要**: 现有 RPKI 体系中, RPKI 资料库与 RP 服务器之间的数据同步使用开源工具 Rsync, 但由于 RPKI 体系中证书数据结构的特殊性, 使用 Rsync 进行数据的同步不仅效率低下, 而且 Rsync 会消耗过多的系统资源, 从而使整个 RPKI 体系遭遇潜在的安全风险. 因此, IETF 针对 RPKI 资料库数据特征, 提出增量同步 Delta 协议以替代 Rsync 在 RPKI 中的作用. 本文详细介绍了 Delta 协议的工作逻辑和机制, 从安全性和高效性两方面将之与 Rsync 进行全面对比, 并使用 Promela 语言构建 Delta 协议模型, 借助形式化验证工具 SPIN 对该模型进行验证, 从而证明该协议具备较高的协议安全性和稳定性. 最后, 本文给出 Delta 协议的实现结构.

**关键词**: RPKI; Delta; SPIN; 增量同步; 形式化检测; 模型检查; 协议安全性

引用格式: 司昊林,马迪,毛伟,王伟,邵晴.RPKI 增量同步 Delta 协议的形式化检测与实现.计算机系统应用,2018,27(11):1–8. http://www.c-s-a.org.cn/1003-3254/6562.html

## Formal Verification and Implementation of RPKI Incremental Synchronous Delta Protocol

SI Hao-Lin[1,2,3], MA Di[2], MAO Wei[2,3], WANG Wei[2], SHAO Qing[3]

[1](Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China)
[2](University of Chinese Academy of Sciences, Beijing 100049, China)
[3](ZDNS Co. Ltd., Beijing 100190, China)

**Abstract**: In the existing RPKI system, the open source tool Rsync is used to synchronize data between the RPKI databases and the RP servers. However, due to the particularity of the certificate data structure in the RPKI system, data synchronization using Rsync not only is inefficient, but also consumes too much system resources, so that the entire RPKI system suffers potential security risks. Therefore, the IETF proposes an incremental synchronization Delta protocol to replace Rsync's role in RPKI. This paper introduces the working logic and mechanism of Delta protocol in detail, compares it with Rsync in terms of security and efficiency, constructs the Delta protocol model by using Promela language, and verifies the model with formal verification tool SPIN. It proves that the protocol has high protocol security and stability. Finally, this paper presents the Delta protocol implementation structure, in order to offer reference and communication.

**Key words**: RPKI; Delta; SPIN; incremental synchronization; formal verification; model checking; protocol security

## 1　RPKI 原理简介

为了解决 BGP 路由劫持的网络安全隐患, 互联网工程任务组 IETF (Internet Engineering Task Force) 提出了 RPKI (Resource Public Key Infrastructure)[1]. RPKI 解决此类隐患的基本思路是: 构建一整套公钥证书体系 PKI (Public Key Infrastructure) 对互联网码号资

① 收稿时间: 2018-02-09; 修改时间: 2018-03-07; 采用时间: 2018-03-14; csa 在线出版时间: 2018-09-30

源 INR (Internet Number Resource), 包括 IP 地址前缀和 AS 号的所有权 (分配) 和使用权 (路由起源通告) 进行验证, 通过验证的结果指示边境路由器是否接受收到的路由起源通告以修改自己的路由信息.

如图 1 所示的 RPKI 体系结构中, 顶级 CA (Certificate Authority) 如 IANA、RIR、NIR、ISP 等在资源分配的过程中, 使用 CA 证书签发一系列用于标识资源所属关系的认证证书, 其中 EE 证书主要用于对路由源授权 (Route Origin Authorization, ROA) 进行验证, 从而确认某 AS 是否可以发起 ASN 与地址段相匹配的合法路由起源通告. 该体系结构中的 RP (Relying Party) 将从资料库同步到的证书构建成数据链表, 使用 OpenSSL 对其进行验证, 验证的结果缓存于本地数据库中, 再通过 RTR (Relying party To Router) 协议向路由器提供查询, 边界路由器通过向 RP 服务器发起查询以确认自身收到的路由源通告是否合法, 从而过滤由错误配置或恶意攻击等生成的非法路由起源通告, 避免路由劫持的发生[2,3].

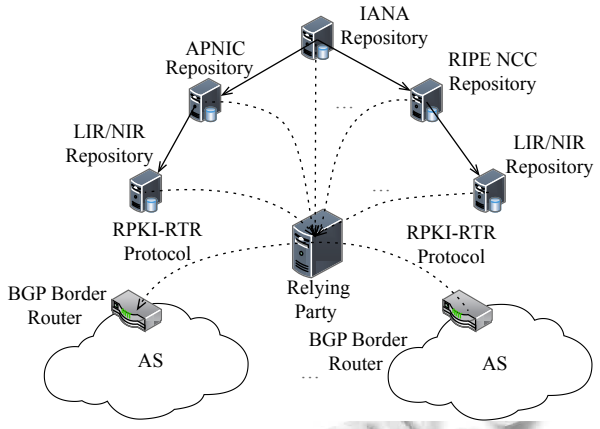


图 1 RPKI 体系架构

## 2 Delta 协议

### 2.1 Delta 协议简介

由于 Rsync 在一般文件的同步过程中表现优异, IETF 在 RPKI 提出初期, 建议使用 Rsync 作为 RP 和资料库之间的同步工具[4]. 但由于 RPKI 数据资料的结构特殊性, 使用 Rsync 进行数据会存在明显缺陷和安全隐患[5].

为了解决 RPKI 在使用 Rsync 过程中存在的缺陷和隐患, IETF 在 2015 年 2 月发布了 RRDP (RPKI Repository Delta Protocol, 简称 Delta 协议) 草案. 经过 10 个版本的修订, 最终于 2017 年 7 月形成标准协议 RFC 8182[6].

相较于 Rsync, Delta 协议具有以下显著特征:

(1) RPKI 资料库需要生成三个文件: 用以发布更新信息的 Notification 更新通告文件、用以发布大块证书打包数据的 Snapshot 快照文件和用以实现增量同步的 Delta 文件, 通过上述文件对同步过程的动态协调, 使 RP 服务器和 RPKI 资料库之间数据同步的可控性大幅提升.

(2) Delta 协议的同步机制中, RPKI 资料库针对证书文件的数据特性, 通过 Delta 文件实现对文件的精确增量更新, RP 服务器初次运行需执行 Snapshot 快照文件外, 后续的增量更新只涉及微量数据, 迅捷高效.

(3) Delta 协议将资料库的证书同步过程与其他功能模块彻底剥离, 这使得 RP 服务器的验证模块可以从本地直接检索构建验证链所需的证书数据, 验证效率被大幅提升. 此外, 同步至 RP 服务器的各类证书数据亦可以结合其他运行数据进行信息分析和挖掘, 对边界路由器提供与指导路由相关的增值服务.

(4) Delta 协议中, 严格的控制文件校验和全方位的安全考量, 使得 RPKI 与 RP 服务器之间数据同步的安全性得到大幅提升, 同时 RPKI 资料库和 RP 服务器之间的控制文件分发采用 HTTPS (Hyper Text Transfer Protocol over Secure socket layer) 协议, 由于对传输数据进行了加密, 可以有效防止中间人 (Monkey-In-the-Middle) 攻击, 提升协议安全性[7].

### 2.2 Delta 协议工作机制

图 2 用以说明 Delta 协议的工作机制.

Delta 协议工作机制

1) RPKI 资料库需以时间 $t$ 为周期, 循环生成 Notification 文件、Snapshot 文件和 Delta 文件并对其进行维护, 这三种文件都由当前的 Session_ID 属性和文件 Serial 号码唯一标识, 并以 URL 方式发布以供远程获取, Session_ID 本身为一个随机版本 4 的通用统一标识符 UUID (Universally Unique Identifier), 用以对各个文件进行唯一性标识. Notification 文件用于对 RP 服务器发布更新会话发起通告, 文件内容包括当前 Delta 版本属性<Version>、本次会话组合标识属性<Session_ID>和<Serial>, 同时也包括本次会话相关的 Snapshot 和 Delta 文件信息. 资料库生成的三个文件均为.xml 文件.

2) RP 服务器周期性获取 Notification 文件, 并对其内容进行解析. 各 RP 服务器可根据自身工具差异, 选取合适的方式对文件进行解析以获取文件信息, 本文中采用 Python2.7 的 xml.etree.ElementTree 模块对.xml 文件进行解析, 并对文件数据构树以备后续使用.

3) RP 服务器需对 Notification 文件格式进行验证. Delta 协议对资料库生成的三种文件均进行了严格的格式规范, 若有任何格式细节不匹配的文件, 都必须在 Delta 协议执行的流程中被拒绝.
4) Notification 文件中携带的<Version>属性值必须为“1”, 标识当前 Delta 协议版本号.
5) Notification 文件中携带的<Serial>属性值需要和本地已存的“Serial”变量进行比较, 通常若 RP 服务器初次发起更新会话, “Serial”变量的值为初始值, 此时需要直接执行 Snapshot 文件, 至步骤 6), 若<Serial>属性值大于本地“Serial”变量值, 则进行 Delta 增量更新, 至步骤 8), 若“Serial”变量值为非初始值且大于等于<Serial>属性值, 则该 Notification 文件被拒绝, 至步骤 11).
6) RP 服务器通过 Notification 文件中 Snapshot 文件的 URL 获取 Snapshot 文件并验证其文件格式和 HASH 值, Snapshot 文件的 HASH 值负载于 Notification 文件中, Delta 协议中的文件 HASH 值均为 SHA-256 散列的十六进制编码. 还需验证 Snapshot 文件的<Session_ID>和<Serial>属性值是否与 Notification 匹配, 若上述三步验证中的任何一个验证失败, 将导致此 Snapshot 文件被拒绝, 至步骤 11).
7) 执行 Snapshot 文件, 获取证书数据, 至步骤 11).
8) 通常一个 Notification 文件会包含多个 Delta 文件信息, 需要对多个 Delta 文件依照<Serial>属性值增序构建文件链表 DeltaFL, 并对 DeltaFL 依次验证和执行.
9) 验证 Delta 文件的文件格式、文件 HASH 值和<Session_ID>、<Serial>属性值, 任何一步验证失败都将导致此 Delta 文件被拒绝, 并执行 Snapshot 文件, 至步骤 6).
10) 执行 Delta 文件, 获取证书数据, 若 DeltaFL 为空, 至步骤 11), 若不为空, 则至步骤 8).
11) 周期循环, 至步骤 2).

### 2.3 Delta 协议的形式化检测

Delta 协议的各文件验证过程层叠环扣, 逻辑较为复杂, 为了准确且全面地说明 Delta 协议及其实现程序的协议正确性 (协议正确性通常指: 不存在违背断言 (assertion) 的情况、不存在不可达 (unreachable) 状态、不存在死锁 (deadlock)、可以完全覆盖定义的 LTL (Linear Temporal Logic) 公式等安全特性)[8], 下文将借助用来验证协议或系统逻辑一致性的工具 SPIN 对 Delta 协议进行形式化检测, 以说明 Delta 具备较高的安全特性.

形式化方法是验证协议一致性和提高软件质量而经常使用的重要方法, 在所有形式化方法中, 模型检查是一种用于验证有穷状态系统模态/命题性质的自动验证技术, 最早由 Clarke 和 Emerson 等[9]提出, 已经被成功地应用于计算机硬件、通信协议、控制系统和安全认证协议等方面的分析和验证中, 它的思想是检验系统中所有可能的状态, 验证这些状态是否满足某些性质. 如果不满足, 将产生一个反例以说明这种情况. 给定系统或协议模型 M 及 LTL 公式 $\phi$, 检测是否有 $L(A_\varphi) \subseteq \text{model}(\varphi)$ 的步骤如算法 1.

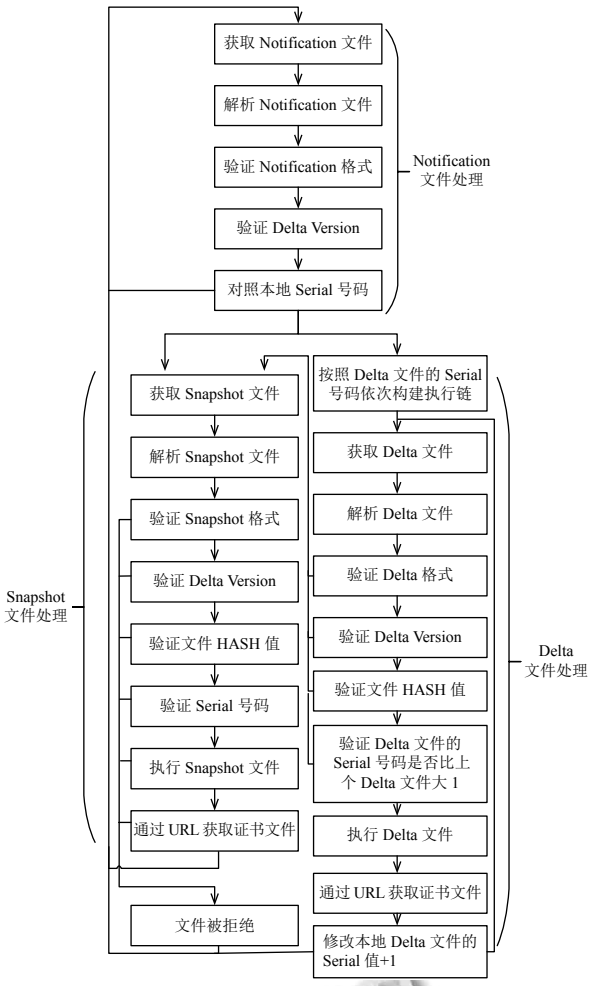


图 2 Delta 协议工作机制示意图

算法 1. 形式化检测算法

1) 构造自动机$A_{\neg\varphi}$, 其对应公式$\neg\varphi$;
2) 计算$A_{M,\varphi}=M\times A_{\neg\varphi}$使得$L(A_{M,\varphi})=L(M)\cap L(A_{\neg\varphi})$;
3) 判定$L(A_{M,\varphi})$是否为空, 也就是$A_{M,\varphi}$不接受任何输入.

SPIN (Simple Promela Interpreter) 是一款适合进行协议一致性检查的分析检测工具, 由贝尔实验室的形式化方法与验证小组开发, SPIN 优秀的算法设计和有效的检测能力使其荣获由 ACM (Association for Computing Machinery) 授予的软件系统奖 (software systems award), 其他获得此殊荣的还有 Unix、TCP/IP、Tcl/Tk、Java、WWW 等[10]. 如图 3 所示的 SPIN 验证流程, SPIN 可以接受由 Promela 建模语言构筑的协议或系统模型, 模型通常由消息通道、变量和进程进行描述. SPIN 会将模型中构筑的进程翻译为有限自动机, 并对这些自动机进行异步积运算得到优先自动机 A, 同时 LTL 公式会被取反转换为自动机 B, 再将自

动机 A 和 B 进行同步积运算得到自动机 C, SPIN 将使用内嵌的搜索算法对 C 进行穷尽搜索[11], 搜索过程可以通过 SPIN 独有的 On-the-fly 技术以及偏序简化技术对状态空间进行简化, 搜索完毕的自动机 C 若其接受的语言为空, 则表示系统满足 LTL 描述的属性, 反之则不满足[12].

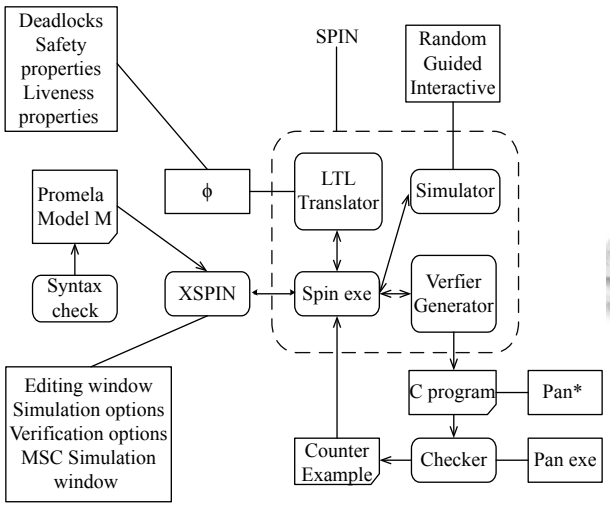


图 3 SPIN 模拟与验证结构流程图

(1) Delta 协议的语言和有限状态机描述

定义一个四元组文法 $G$: $G=(V,T,P,S)$, 其中, $V$ 是变量集合, $\forall A\in V$, $A$ 叫做一个语法变量; $T$ 是终极符的集合, $T$ 中的字符是语言的句子中出现的字符, $V\cap T=\emptyset$; $P$ 是产生式的集合, $P$ 中的元素具有形式 $\alpha\to\beta;S\in V$, 文法 $G$ 的开始符号.

因此, Delta 协议的验证逻辑可以使用此语法 $G$ 进行表述:

$$G_D=(\{S,A,B,C\},\{a,b,c\},\{S\to aA,A\to aA|aB|aC,B\to aA|b,C\to aC|aB|c\},S)$$

则使用文法$G_D$可生成如下的语言:

| | | |
|---|---|---|
| $S$ | $\Rightarrow aA$ | 使用产生式$S\to aA$ |
| | $\Rightarrow aaA$ | 使用产生式$A\to aA$ |
| | $\Rightarrow aaaB$ | 使用产生式$A\to aB$ |
| | $\Rightarrow aaaaA$ | 使用产生式$B\to aA$ |
| | $\Rightarrow aaaaB$ | 使用产生式$A\to aB$ |
| | $\Rightarrow aaaab$ | 使用产生式$B\to b$ |
| $S$ | $\Rightarrow aA$ | 使用产生式$S\to aA$ |
| | $\Rightarrow aaC$ | 使用产生式$A\to aC$ |
| | $\Rightarrow aaaC$ | 使用产生式$C\to aC$ |
| | $\Rightarrow aaaB$ | 使用产生式$C\to aB$ |
| | $\Rightarrow aaab$ | 使用产生式$B\to b$ |
| | $\Rightarrow aaac$ | 使用产生式$C\to c$ |

文法 $G_D$ 可产生的语言为: $\{a^nb|n\geq 1\}\cup\{a^nc|n\geq 1\}$; 可接受或识别上述语言的有限状态机 DeltaState 如图 4.

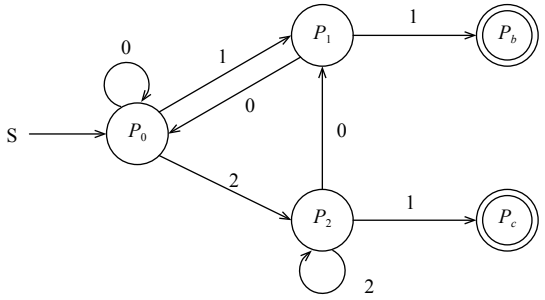


图 4 Delta 协议有限自动状态机

表 1 Delta 对应的自动机转移状态 DeltaState

| 状态 | 含义 |
|---|---|
| $P_0$ | 获取 Notification 文件并验证 |
| $P_1$ | N 文件验证通过并获取 Snapshot 文件进行验证 |
| $P_2$ | N 文件验证通过并获取 Delta 文件验证、构建文件链 DeltaFL |
| $P_b$ | S 文件验证通过并对 S 文件进行执行, 为接受状态 |
| $P_c$ | D 文件验证通过并对 D 文件进行执行, 为接受状态 |

有限自动状态机 DeltaState 初始进入状态$P_0$, $P_0$状态获取 Notification 文件并对其进行验证, 根据验证结果进行状态转移: 若验证结果为 0, 则重复自身状态, 重新获取 Notification 文件并验证; 若验证结果为 1, 则转移至协议状态$P_1$; 若验证结果为 2, 则转移至协议状态$P_2$. $P_1$状态, 获取 Snapshot 文件并验证, 根据验证结果进行状态转移: 若验证结果为 0, 该 Snapshot 文件被拒绝, 转移至协议状态$P_0$; 若验证结果为 1, 则转移至协议状态$P_b$. $P_2$状态, 获取 Delta 文件并构建 DeltaFL 文件链表, 进行验证, 根据验证结果进行状态转移: 若验证结果为 0, 则转移至协议状态$P_1$; 若验证结果为 2, 则转移至自身继续完成 DeltaFL 文件链表的构建; 若验证结果为 1, 则转移至协议状态$P_c$; 协议状态$P_b$和协议状态$P_c$均为可接受状态, $P_b$协议状态表示 Snapshot 文件通过验证且被执行, $P_c$协议状态则表示 DeltaFL 文件链表构建完成且被执行.

(2) Delta 协议模型的 Promela 描述

SPIN 需要接受由 Promela 语言进行描述的协议或系统模型, 并对其进行转化和验证. Promela 语言是一种用来描述并发系统 (concurrent systems) 的模型语言 (modelling language), 可以使用 Promela 语言模拟和创

建进程, 表述变量, 通过进程间信息传输等对模型进行描述[13]. 使用 Promela 语言对 Delta 协议进行如下描述.

Delta 协议的 Promela 描述

```
1.  chan notifFile=[1]of{byte};
                …                          /*定义全局消息通道*/
2.  chan deltaData=[1]of{byte};
3.  active proctype Library(){
4.  byte nF=1, sF=1, sD=1, dF=1, dD=1;
5.  byte rubbish;
6.  do                                     /*Library 进程, 模拟文
7.  ::notifFile!nF                         件生成*/
                …
8.  ::deltaData!dD
9.  od}
10. active proctype RP(){
11. byte getNoti;
                …
12. byte deltaState;
13. do
14. ::notifFile?getNoti;                   /*RP 进程变量定义和
15. ::(getNoti==0)->goto continue          状态转移*/
16. if
17. fi
18. ::(notiState==0)->goto refuse
                …
19. ::(notiState==2)->goto proDelta
20. proSnapsh:
                …
21. ::snapshFile?getSnapsh;
22. ::(getSnapsh==0)->goto continue
23. if
                …                          /*RP 进程中的
24. fi                                     Snapshot 处理状态*/
25. ::(snapshState==0)->goto refuse
26. progress1:
    snapshData?getSnapshData
27. ::(getSnapshData==0)->goto continue
28. goto continue
29. proDelta:
                …
30. ::deltaFile?getDelta;
31. ::(getDelta==0)->goto continue
32. if                                     /*RP 进程中的 Delta 处
                …                          理状态*/
33. fi
34. ::(deltaState==0)->goto refuse
35. progress2:deltaData?getDeltaData
36. ::(getDeltaData==0)->goto continue
37. goto continue
                …
38. refuse:
                …                          /*RP 进程中出错或循
39. continue:                              环转移状态*/
                …
40. od}
```

图 5 为 Promela 模型中各进程间信息传递过程及状态转移图. 在 Promela 模型中, 构建了两个进程 proctype_Library 和 proctype_RP 分别用以模拟 Delta 协议中 RPKI 资料库端和 RP 依赖方的运行状态. proctype_Library 进程对控制文件的生成和数据打包进行了模拟, 此进程为循环进程, 若产生文件生成或数据打包失败则循环, 生成的文件和数据都将被公用全局通道变量 notifFile、snapshFile、snapshData、deltaFile、deltaData 负载以供 proctype_RP 进程获取. proctype_RP 为 Delta 主要的协议逻辑模拟进程, 表 2 中为进程中变量与之对应的模拟状态和模型中取值.

表 2 Delta 协议模型进程内变量

| 进程内变量 | 模拟状态 | 值 |
|---|---|---|
| getNoti | Notification 文件获取状态 | |
| notiState | Notification 文件验证状态 | |
| getSnapsh | Snapshot 文件获取状态 | |
| getSnapshData | Snapshot 数据获取状态 | 0/1 |
| snapshState | Snapshot 文件验证状态 | |
| getDelta | Delta 文件获取状态 | |
| getDeltaData | Delta 数据获取状态 | |
| deltaState | Delta 文件验证状态 | 0/1/2 |

proctype_RP 进程中主要的循环逻辑在 do…od 循环体中, 表 2 中的状态变量则由 if…fi 结构内的语句进行随机的数值变换, 以表述文件验证或数据获取的成功与否, 根据状态数据表述的结果在逻辑处理之间使用 goto 语句进行跳转, 主要的三个处理逻辑部分分别为主循环体中的 Notification 文件处理逻辑、Snapshot 文件处理逻辑 proSnapsh 和 Delta 文件处理逻辑 proDelta. 同时在 Promela 模型中标注了下述语句: ::(1)->progress1: snapshData?getSnapshData 和::(1)->progress2:deltaData?getDeltaData 分别使用模型标记关键字 progress 用于指示 SPIN 在验证过程不允许出现从不执行语句 snapshData?getSnapshData 和 deltaData?getDeltaData 的循环发生, 因为此两条语句所表示的模型意义分别是从 proctype_Library 进程获取 Snapshot 数据和 Delta 数据, 为该验证模型必须可达的“可接受”状态.

(3) Delta 协议模型的 SPIN 验证

图 6 所示是由 Promela 模型生成的 Delta 协议逻辑自动机, 其本质与图 4 自动机相同, 只不过在 Promela 描述中加入了循环用转移状态, 所以略有差异.

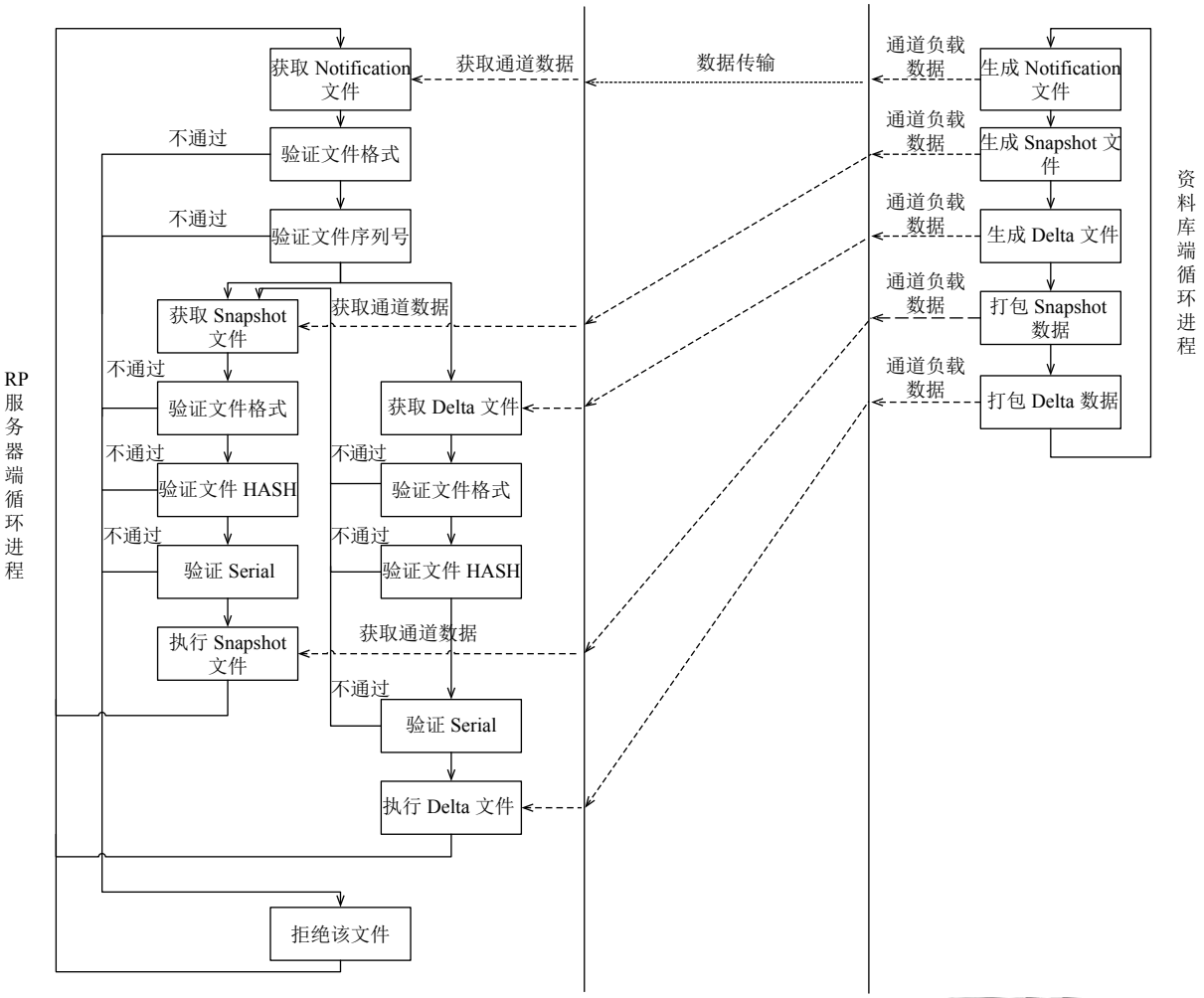


图 5 Delta 协议 Promela 模型

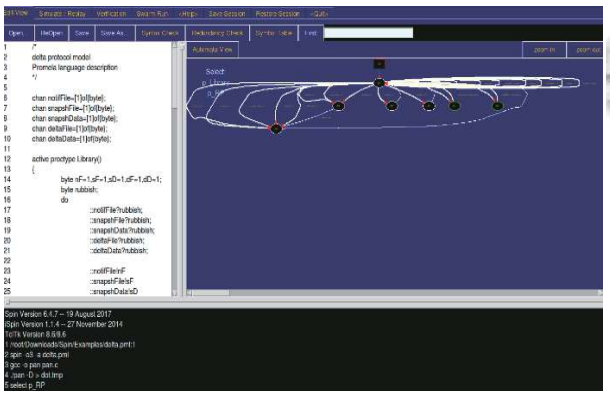

图 6 Promela 模型生成的自动机

图 7 为使用 SPIN 对 Delta 协议的 Promela 模型进行验证的结果, 验证结果表示 Delta 协议不存在“死锁”、“无效循环”等不安全协议特性, 同时其协议逻辑完全可达.

图 8 为 Promela 模型的模拟运行, 共进行 10 000 步模拟运行, 无任何报错, 协议稳定性较高.

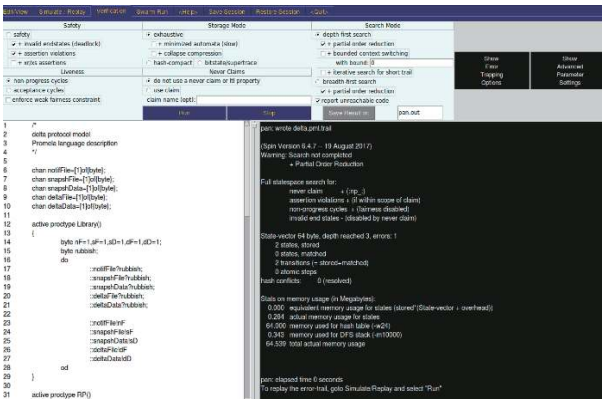

图 7 Promela 模型验证结果

通过上述验证过程, 可以从逻辑层面非常严密地

证明: Delta 协议不存在“死锁”、“无效循环”等不安全协议特性, 同时其协议逻辑完全可达, 具有非常高的协议安全性. 通过模拟运行则可以体现出其具备极高的稳定性.

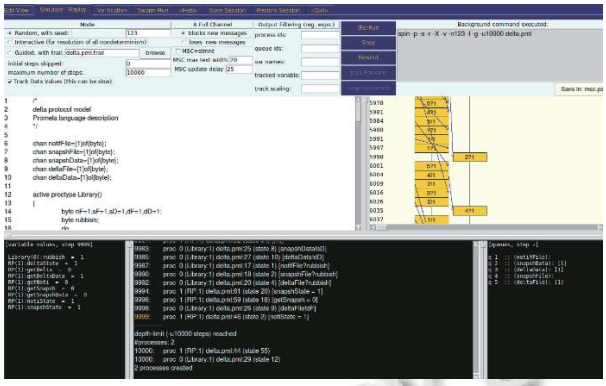

图 8 Promela 模型模拟运行

## 3 Delta 协议实现

Promela 构建的协议模型不仅可以对协议验证进行模拟, 同时由于具备完整的协议结构, 也可以在协议的实现中进行指导. 本文基于 Delta 的 Promela 模型, 使用 Python 对 Delta 协议进行了实现开发. 截止本文撰写, 该 Delta 功能为国内首次实现, 源码已呈现于 GitHub 供开源使用 https://github.com/sihaolin/RPKI-Delta-Protocol. 表 3 为该协议实现的各主要功能函数, 可以从逻辑上完整搭建 Delta 协议的工程架构, 望能对其他有需求的开发者提供参考和帮助.

## 4 总结

通过上文阐述, 可以看出 Delta 协议具有较高协议安全特性, 且其协议逻辑稳定. 相较于 RPKI 体系中早期使用的 Rsync 同步工具, Delta 协议的同步可控性得到大幅提升, 增量更新的方式也使得其更新效率大幅提高, 严密的控制文件格式验证和 HTTPS 协议对传输数据的加密也使得数据同步的安全性得到保障, Delta 协议对资料库服务器更少的资源占用则使得服务器在遭受 DDOS 攻击时具有更高的抵御力. Delta 协议已经较为成熟, 且具备 RPKI 体系所需的优秀特性, 在未来一段时间内将会完全替代 Rsync, 成为组成 RPKI 体系的重要组件.

表 3 Delta 实现的主要功能函数

| Delta 协议主要功能函数 | 功能示意 |
|---|---|
| delta_Read_Conf() | 解析配置文件, 获取控制文件的 CA 分发点 |
| obtain_File(current_Node) | 获取当前控制文件 |
| obtain_H_Sha256(file_Name) | 生成文件的 HASH 值 |
| validate_Notifica_File_Fo() | 验证 Notification 格式 |
| validate_Snapsh_File_Fo() | 验证 Snapshot 格式 |
| validate_Delta_File_Format() | 验证 Delta 文件格式 |
| parse_Notification_File() | 解析 Notification 文件, 获取文件负载的信息 |
| parse_Snapshot_File() | 解析 Snapshot 文件, 构建元素树 s_E, 并返回数据 |
| parse_Delta_File() | 解析 Snapshot 文件, 构建元素树 d_E, 并返回数据 |
| rsync_File(URI) | 通过 URI 远程获取证书数据 |
| processing_Snap_File(s_E) | 传入元素树 s_E, 执行 Snapshot 文件 |
| processing_Delta_File(d_E) | 传入元素树 d_E, 执行 Delta 文件 |
| main() | 主体循环, 调用其他功能函数, 构建各个文件验证的主要协议逻辑, 并获取数据 |

## 参考文献


1 马迪, 沈烁. 基于本地信任锚点管理的 RPKI 安全运行机制研究. 电信科学, 2013, 29(9): 55–59. [doi: 10.3969/j.issn.1000-0801.2013.09.010]

2 贾佳, 延志伟, 耿光刚, 等. BGP 路由泄露研究. 网络与信息安全学报, 2016, 2(8): 54–61.

3 Ballani H, Francis P, Zhang XY. A study of prefix hijacking and interception in the internet. Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications. Kyoto, Japan. 2007. 265–276. [doi: 10.1145/1282427.1282411]

4 许圣明, 马迪, 毛伟, 等. 基于有序哈希树的 RPKI 资料库数据同步方法. 计算机系统应用, 2016, 25(6): 141–146. [doi: 10.15888/j.cnki.csa.005203]

5 刘晓伟, 延志伟, 耿光刚, 等. RPKI 中 CA 资源分配风险及防护技术. 计算机系统应用, 2016, 25(8): 16–22. [doi: 10.15888/j.cnki.csa.005313]

6 王娜, 杜学绘, 王文娟, 等. 边界网关协议安全研究综述. 计算机学报, 2017, 40(7): 1626–1648.

7 肖美华, 薛锦云. 基于 SPIN/Promela 的并发系统验证. 计算机科学, 2004, 31(8): 201–203, 208. [doi: 10.3969/j.issn.1002-137X.2004.08.059]

8 韩德帅, 杨启亮, 邢建春. 一种软件自适应 UML 建模及其形式化验证方法. 软件学报, 2015, 26(4): 730–746. [doi: 10.13328/j.cnki.jos.004758]

9 郭伟, 缪力, 张大方, 等. 基于 Spin 的 UML 状态图模型检

查的设计与实现. 计算机工程与应用, 2008, 44(10): 43–47. [doi: 10.3778/j.issn.1002-8331.2008.10.013]

10 Yang QL, Lv J, Tao XP, *et al*. Fuzzy self-adaptation of mission-critical software under uncertainty. Journal of Computer Science and Technology, 2013, 28(1): 165–187. [doi: 10.1007/s11390-013-1321-9]

11 Holzmann G J. The model checker SPIN. IEEE Transactions on Software Engineering, 1997, 23(5): 279–295. [doi: 10.1109/32.588521]

12 Yang WH, Xu C, Liu YP, *et al*. Verifying self-adaptive applications suffering uncertainty. Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering. Vasteras, Sweden. 2014. 199–210. [doi: 10.1145/2642937.2642999]

13 Huston G, Michaelson G. RFC 6483 - Validation of route origination using the resource certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs). BMC Public Health, 2012, 11(1): 1–6.