(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 116684247 A
(43)申请公布日 2023.09.01

(21)申请号 202310710233.X

(22)申请日 2023.06.15

(71)申请人 厦门大学
地址 361005 福建省厦门市思明区思明南路422号

(72)发明人 向乔 秦秋月 徐惠三 方星 舒继武

(74)专利代理机构 北京庚致知识产权代理事务所(特殊普通合伙) 11807
专利代理师 韩德凯

(51)Int.Cl.
H04L 41/06(2022.01)
H04L 41/14(2022.01)

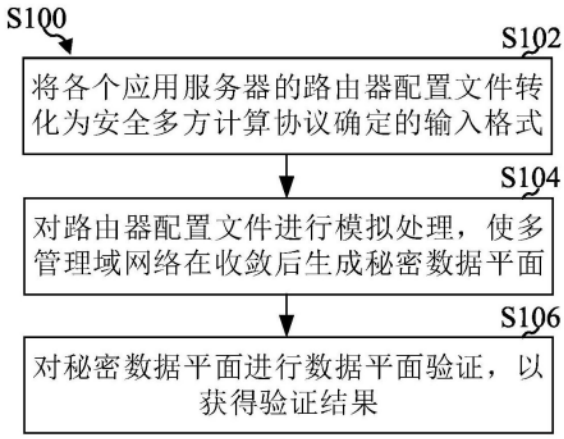
权利要求书2页 说明书11页 附图12页

(54)发明名称

多管理域网络验证方法及系统

(57)摘要

本公开提供了一种多管理域网络验证方法，包括:将各个应用服务器的路由器配置文件转化为安全多方计算协议确定的输入格式,以保证各个应用服务器之间的兼容性;对满足输入格式的路由器配置文件进行模拟处理,以使得多管理域网络在收敛后生成秘密数据平面;以及对秘密数据平面进行数据平面验证,以获得关于多管理域网络的各个属性的验证结果。本公开还提供一种多管理域网络验证系统。



CN 116684247 A

1.一种多管理域网络验证方法,其特征在于,包括:

将各个应用服务器的路由器配置文件转化为安全多方计算协议确定的输入格式,以保证各个所述应用服务器之间的兼容性;

对满足所述输入格式的所述路由器配置文件进行模拟处理,以使得多管理域网络在收敛后生成秘密数据平面;以及

对所述秘密数据平面进行数据平面验证,以获得关于所述多管理域网络的各个属性的验证结果。

2.根据权利要求1所述的多管理域网络验证方法,其特征在于,所述将各个应用服务器的路由器配置文件转化为安全多方计算协议确定的输入格式,包括:

将各个所述应用服务器的路由器配置文件转化为相同的中间表示;

将所述中间表示转化为所述安全多方计算协议确定的输入格式。

3.根据权利要求2所述的多管理域网络验证方法,其特征在于,所述安全多方计算协议确定的输入格式包括:域间路由信息表、所述应用服务器的路由器配置信息、所述应用服务器的路由器节点更新状态和路由信息更新列表,其中

所述域间路由信息表至少用于存储有路线数据,其中所述路线数据至少包括路由宣告的前缀、下一路由器节点的网络协议地址、所经过的其余路由器节点、当前的所述路由器节点的本地优先级、变更路由器信息;以及

所述应用服务器的路由器配置信息用于表征当前的所述路由器节点与相邻路由器节点之间的关系,其中所述路由器配置信息至少包括所述相邻路由器节点的网络协议地址、当前的所述路由器节点的应用服务器编号、所述相邻路由器节点的应用服务器编号、当前的所述路由器节点对应的接口网络协议地址、当前的所述路由器节点对应的输入输出策略。

4.根据权利要求1所述的多管理域网络验证方法,其特征在于,所述对满足所述输入格式的所述路由器配置文件进行模拟处理,以使得多管理域网络在收敛后生成秘密数据平面,包括:

对各个所述应用服务器中路由器节点的域间路由信息表进行初始化,使得各个所述域间路由信息表中仅存储有原始路由器节点;

响应于任一所述域间路由信息表的更新信息,对所述更新信息对应的所述路由器节点的对等路由器节点发送路由通知;以及

控制各个所述路由器节点执行相应的所述路由通告,直至所述多管理域网络收敛并生成所述秘密数据平面。

5.根据权利要求4所述的多管理域网络验证方法,其特征在于,所述响应于任一所述域间路由信息表的更新信息,对所述更新信息对应的所述路由器节点的对等路由器节点发送路由通知,至少包括:

利用出口路线函数对所述路由器节点的导出过滤器的动作进行建模,以将所述更新信息转换为出口数据;以及

利用入口路线函数对所述对等路由器节点上的导入过滤器进行建模,并修改所述出口数据。

6.根据权利要求1所述的多管理域网络验证方法,其特征在于,所述对所述秘密数据平

面进行数据平面验证,以获得关于所述多管理域网络的各个属性的验证结果,包括:

控制目的路由器节点执行数据无关搜索操作,判断所述目的路由器节点对应的有效路径中是否存在到达入口路由器节点的路径;以及

响应于存在到达所述入口路由器节点的路径,确定所述入口路由器节点至所述目的路由器节点满足可达性,以获得关于所述多管理域网络的可达性属性的验证结果。

7.根据权利要求1所述的多管理域网络验证方法,其特征在于,还包括:

在对所述秘密数据平面进行数据平面验证时,将各个所述应用服务器的私有值进行加密,以防止所述私有值被读取。

8.根据权利要求1所述的多管理域网络验证方法,其特征在于,在所述对所述秘密数据平面进行数据平面验证,以获得关于所述多管理域网络的各个属性的验证结果之后,包括:

将所述验证结果同步至各个所述应用服务器,以使得各个所述应用服务器获得所述多管理域网络所满足的属性。

9.根据权利要求1所述的多管理域网络验证方法,其特征在于,在所述将各个应用服务器的路由器配置文件转化为安全多方计算协议确定的输入格式,以保证各个所述应用服务器之间的兼容性之前,包括:

将各个所述应用服务器的路由器节点与代理服务器进行连接和认证,并将各个所述路由器节点的路由器配置信息发送至所述代理服务器。

10.一种多管理域网络验证系统,其特征在于,包括:

配置解析模块,用于将各个应用服务器的路由器配置文件转化为安全多方计算协议确定的输入格式,以保证各个所述应用服务器之间的兼容性;

路由协议模拟模块,用于对满足所述输入格式的所述路由器配置文件进行模拟处理,以使得多管理域网络在收敛后生成秘密数据平面;以及

数据平面验证模块,用于对所述秘密数据平面进行数据平面验证,以获得关于所述多管理域网络的各个属性的验证结果。

多管理域网络验证方法及系统

技术领域

[0001] 本公开涉及网络安全技术领域,特别涉及一种多管理域网络验证方法及系统。

背景技术

[0002] 网络验证是一种自动化网络故障检测技术,它通过检查网络设备的数据平面(即路由表、访问控制表和流表等报文转发规则表)和控制平面(即设备配置文件)判断网络行为是否符合网络正确性要求,例如路由无环、路由无黑洞和可达性等。图1为相关技术的数据平面网络验证示例图,参通过检查路由器A和路由器B的转发信息库,数据平面验证系统可以判断出网络中目的IP地址为10.0.0.0的报文将在A和B之间陷入路由回路,而目的IP地址为10.0.0.1的将被B丢弃,从而进入黑洞,二者均无法正常送达目的主机Host 2,进而导致网络故障。图2为相关技术的控制平面网络验证的示例图,路由器A和路由器B之间通过边界网络协议BGP连接,控制平面验证系统通过检查B的配置文件可以判断出B不会通往目的IP前缀10.0.0.0/31的BGP路由宣告给A,以致A无法将任何目的IP地址属于10.0.0.0/31的报文送达目的主机Host 2,进而导致网络故障。

[0003] 相关技术中的网络故障诊断工具主要分为以下三类:第一类研究通过构建与实际网络相同的仿真环境对配置文件进行仿真,然后使用数据平面验证工具检查仿真得到的数据平面,以达到检测配置文件正确性的目的;第二类研究通过将网络配置与网络拓扑融合,构建特殊的抽象图模型,并通过在抽象图上寻找满足特定条件的路径来验证配置文件是否满足网络正确性;第三类研究使用形式化方法,利用数据逻辑对网络设备的配置文件与需要检查的正确性要求进行逻辑建模,并通过检查所构建的逻辑模型的可满足性,判断相应网络配置是否满足相应的正确性。

[0004] 但是,第一类研究的局限性在于有效性,不能应对网络延迟等各种真实场景所产生的问题;第二类研究的局限性在于有效性和使用范围,该类验证工具无法验证许多常见的网络协议配置。第三类研究的局限性在于规模性较差,对于一些简单的网络配置进行网络可达性验证时,还需要较长的验证时间。另外,无论是数据平面还是控制平面,相关技术中的网络验证工具服务的主要对象都是单管理域网络,对于多管理域网络故障的排查还主要依赖于不同管理域的网络管理员之间的沟通,这样的验证方式效率较低。

发明内容

[0005] 为了解决上述的至少一个问题,本公开提供了一种多管理域网络验证方法及系统。

[0006] 根据本公开的一个方面提出了这样一种多管理域网络验证方法,包括:将各个应用服务器的路由器配置文件转化为安全多方计算协议确定的输入格式,以保证各个所述应用服务器之间的兼容性;对满足所述输入格式的所述路由器配置文件进行模拟处理,以使得多管理域网络在收敛后生成秘密数据平面;以及对所述秘密数据平面进行数据平面验证,以获得关于所述多管理域网络的各个属性的验证结果。

[0007] 在一些实施方式中，所述将各个应用服务器的路由器配置文件转化为安全多方计算协议确定的输入格式，包括：将各个所述应用服务器的路由器配置文件转化为相同的中间表示；将所述中间表示转化为所述安全多方计算协议确定的输入格式。

[0008] 在一些实施方式中，所述安全多方计算协议确定的输入格式包括：域间路由信息表、所述应用服务器的路由器配置信息、所述应用服务器的路由器节点更新状态和路由信息更新列表，其中所述域间路由信息表至少用于存储有路线数据，其中所述路线数据至少包括路由宣告的前缀、下一路由器节点的网络协议地址、所经过的其余路由器节点、当前的所述路由器节点的本地优先级、变更路由器信息；以及所述应用服务器的路由器配置信息用于表征当前的所述路由器节点与相邻路由器节点之间的关系，其中所述路由器配置信息至少包括所述相邻路由器节点的网络协议地址、当前的所述路由器节点的应用服务器编号、所述相邻路由器节点的应用服务器编号、当前的所述路由器节点对应的接口网络协议地址、当前的所述路由器节点对应的输入输出策略。

[0009] 在一些实施方式中，所述对满足所述输入格式的所述路由器配置文件进行模拟处理，以使得多管理域网络在收敛后生成秘密数据平面，包括：对各个所述应用服务器中路由器节点的域间路由信息表进行初始化，使得各个所述域间路由信息表中仅存储有原始路由器节点；响应于任一所述域间路由信息表的更新信息，对所述更新信息对应的所述路由器节点的对等路由器节点发送路由通知；以及控制各个所述路由器节点执行相应的所述路由通告，直至所述多管理域网络收敛并生成所述秘密数据平面。

[0010] 在一些实施方式中，所述响应于任一所述域间路由信息表的更新信息，对所述更新信息对应的所述路由器节点的对等路由器节点发送路由通知，至少包括：利用出口路线函数对所述路由器节点的导出过滤器的动作进行建模，以将所述更新信息转换为出口数据；以及利用入口路线函数对所述对等路由器节点上的导入过滤器进行建模，并修改所述出口数据。

[0011] 在一些实施方式中，所述对所述秘密数据平面进行数据平面验证，以获得关于所述多管理域网络的各个属性的验证结果，包括：控制目的路由器节点执行数据无关搜索操作，判断所述目的路由器节点对应的有效路径中是否存在到达入口路由器节点的路径；以及响应于存在到达所述入口路由器节点的路径，确定所述入口路由器节点至所述目的路由器节点满足可达性，以获得关于所述多管理域网络的可达性属性的验证结果。

[0012] 在一些实施方式中，还包括：在对所述秘密数据平面进行数据平面验证时，将各个所述应用服务器的私有值进行加密，以防止所述私有值被读取。

[0013] 在一些实施方式中，在所述对所述秘密数据平面进行数据平面验证，以获得关于所述多管理域网络的各个属性的验证结果之后，包括：将所述验证结果同步至各个所述应用服务器，以使得各个所述应用服务器获得所述多管理域网络所满足的属性。

[0014] 在一些实施方式中，在所述将各个应用服务器的路由器配置文件转化为安全多方计算协议确定的输入格式，以保证各个所述应用服务器之间的兼容性之前，包括：将各个所述应用服务器的路由器节点与代理服务器进行连接和认证，并将各个所述路由器节点的路由器配置信息发送至所述代理服务器。

[0015] 根据本公开的另一个方面提供了这样一种多管理域网络验证系统，包括：配置解析模块，用于将各个应用服务器的路由器配置文件转化为安全多方计算协议确定的输入格

式,以保证各个所述应用服务器之间的兼容性;路由协议模拟模块,用于对满足所述输入格式的所述路由器配置文件进行模拟处理,以使得多管理域网络在收敛后生成秘密数据平面;以及数据平面验证模块,用于对所述秘密数据平面进行数据平面验证,以获得关于所述多管理域网络的各个属性的验证结果。

附图说明

[0016] 附图示出了本公开的示例性实施方式,并与其说明一起用于解释本公开的原理,其中包括了这些附图以提供对本公开的进一步理解,并且附图包括在本说明书中并构成本说明书的一部分。

[0017] 图1为相关技术的数据平面网络验证示例图。

[0018] 图2为相关技术的控制平面网络验证的示例图。

[0019] 图3为本公开示例性实施方式的多管理域网络验证方法框图。

[0020] 图4为本公开示例性实施方式的多管理域网络验证系统示意图。

[0021] 图5为本公开示例性实施方式的InCV云服务器架构示意图。

[0022] 图6为本公开示例性实施方式的配置文件厂商中立化示意图。

[0023] 图7至14依次为本公开示例性实施方式的DO-Simulation算法1至8的代码示意图。

[0024] 图15为本公开示例性实施方式的数据无关算法的代码示意图。

[0025] 图16为本公开示例性实施方式的不同规模网络的时间开销对比图。

[0026] 图17为本公开示例性实施方式的不同规模网络的通信轮次与全局数量对比图。

[0027] 图18为本公开示例性实施方式的InCV与未采用FASTPLANE优化版本的对比图。

具体实施方式

[0028] 下面结合附图和实施方式对本公开作进一步的详细说明。可以理解的是,此处所描述的具体实施方式仅用于解释相关内容,而非对本公开的限定。另外还需要说明的是,为了便于描述,附图中仅示出了与本公开相关的部分。

[0029] 需要说明的是,在不冲突的情况下,本公开中的实施方式及实施方式中的特征可以相互组合。下面将参考附图并结合实施方式来详细说明本公开的技术方案。

[0030] 除非另有说明,否则示出的示例性实施方式/实施例将被理解为提供可以在实践中实施本公开的技术构思的一些方式的各种细节的示例性特征。因此,除非另有说明,否则在不脱离本公开的技术构思的情况下,各种实施方式/实施例的特征可以另外地组合、分离、互换和/或重新布置。

[0031] 本文使用的术语是为了描述具体实施例的目的,而不是限制性的。如这里所使用的,除非上下文另外清楚地指出,否则单数形式“一个(种、者)”和“所述(该)”也意图包括复数形式。此外,当在本说明书中使用术语“包含”和/或“包括”以及它们的变型时,说明存在所陈述的特征、整体、步骤、操作、部件、组件和/或它们的组,但不排除存在或附加一个或更多个其它特征、整体、步骤、操作、部件、组件和/或它们的组。还要注意的是,如这里使用的,术语“基本上”、“大约”和其它类似的术语被用作近似术语而不用作程度术语,如此,它们被用来解释本领域普通技术人员将认识到的测量值、计算值和/或提供的值的固有偏差。

[0032] 图3为本公开示例性实施方式的多管理域网络验证方法框图。下面结合图3对多管

理域网络验证方法S100的各个步骤进行详细说明。

[0033] 如图3所示,根据本公开的一个方面提出了这样一种多管理域网络验证方法S100,包括:

[0034] 步骤S102,将各个应用服务器的路由器配置文件转化为安全多方计算协议确定的输入格式,以保证各个应用服务器之间的兼容性。

[0035] 步骤S104,对满足输入格式的路由器配置文件进行模拟处理,以使得多管理域网络在收敛后生成秘密数据平面。

[0036] 步骤S106,对秘密数据平面进行数据平面验证,以获得关于多管理域网络的各个属性的验证结果。

[0037] 路由器配置文件用于表征应用服务器中路由节点的配置信息。安全多方计算(Secure Muti-party Computation,SMPC)协议以密码学算法保证了在数据平面验证的过程中的数据安全性,以避免数据的泄露和伪造。

[0038] 在一些实施方式中,步骤S102的具体实施方式包括:将各个应用服务器的路由器配置文件转化为相同的中间表示;将中间表示转化为安全多方计算协议确定的输入格式。

[0039] 在一些实施方式中,安全多方计算协议确定的输入格式包括:域间路由信息表(Routing Information Base,RIB)、应用服务器的路由器配置信息Config、应用服务器的路由器节点更新状态HasUpdate和路由信息更新列表UpdateList,其中域间路由信息表至少用于存储有路线数据Route,其中路线数据至少包括路由宣告的前缀Prefix、下一路由器节点的网络协议地址NextHop、所经过的其余路由器节点AsPath、当前的路由器节点的本地优先级LocalPref、变更路由器信息IDs;以及应用服务器的路由器配置信息用于表征当前的路由器节点与相邻路由器节点之间的关系,其中路由器配置信息至少包括相邻路由器节点的网络协议地址PeerAddress、当前的路由器节点的应用服务器编号LocalAs、相邻路由器节点的应用服务器编号RemoteAs、当前的路由器节点对应的接口网络协议地址Interface、当前的路由器节点对应的输入输出策略Policy。

[0040] 其中,当前的路由器节点对应的输入输出策略Policy至少包括输入策略ImportPolicy和输出策略ExportPolicy。

[0041] 在一些实施方式中,步骤S104的具体执行步骤包括:对各个应用服务器中路由器节点的域间路由信息表进行初始化,使得各个域间路由信息表中仅存储有原始路由器节点;响应于任一域间路由信息表的更新信息,对更新信息对应的路由器节点的对等路由器节点发送路由通知;以及控制各个路由器节点执行相应的路由通告,直至多管理域网络收敛并生成秘密数据平面。

[0042] 在一些实施方式中,响应于任一域间路由信息表的更新信息,对更新信息对应的路由器节点的对等路由器节点发送路由通知,至少包括:利用出口路线函数对路由器节点的导出过滤器的动作进行建模,以将更新信息转换为出口数据;以及利用入口路线函数对对等路由器节点上的导入过滤器进行建模,并修改所述出口数据。

[0043] 在一些实施方式中,步骤S106的具体执行步骤为:控制目的路由器节点执行数据无关搜索操作,判断目的路由器节点对应的有效路径中是否存在到达入口路由器节点的路径;以及响应于存在到达入口路由器节点的路径,确定入口路由器节点至目的路由器节点满足可达性,以获得关于多管理域网络的可达性属性的验证结果。

[0044] 在一些实施方式中，多管理域网络验证方法S100还包括：在对秘密数据平面进行数据平面验证时，将各个应用服务器的私有值进行加密，以防止私有值被读取。

[0045] 在一些实施方式中，在步骤S106之后，包括：将验证结果同步至各个应用服务器，以使得各个应用服务器获得多管理域网络所满足的属性。

[0046] 在一些实施方式中，在步骤S02之前，包括：将各个应用服务器的路由器节点与代理服务器进行连接和认证，并将各个路由器节点的路由器配置信息发送至代理服务器。

[0047] 图4为本公开示例性实施方式的多管理域网络验证系统示意图。

[0048] 参考图4，本公开的系统可以分为三个串联的模块：配置解析模块、隐私保护下的路由协议模拟模块和隐私保护下的数据平面验证模块。

[0049] 具体地，配置解析模块用于将各个应用服务器的路由器配置文件转化为安全多方计算协议确定的输入格式(即统一格式的SMPC输入)，以保证各个应用服务器之间的兼容性。路由协议模拟模块用于对满足输入格式的路由器配置文件进行模拟处理，以使得多管理域网络在收敛后生成秘密数据平面(即隐私的数据平面RIB)。数据平面验证模块用于对秘密数据平面进行数据平面验证，以获得关于多管理域网络的各个属性的验证结果。

[0050] 在一些实施方式中，数据平面验证模块还会将验证结果以公开的方式返回给所有的参与方，即各个应用服务器。例如AS(Application Server，应用服务器)1、AS2和AS3，各个AS的厂商在此不做限制。

[0051] 具体地，配置解析模块首先负责eBGP(external Border Gateway Protocol，外部边界网关协议)路由器与代理服务器的连接与认证。在完成认证后，每个AS的eBGP路由器可将配置发送至代理服务器，或进行配置的更改。进一步地，配置解析模块将各个应用服务器的路由器配置文件转化为SMPC协议确定的输入格式，以解决不同应用服务器之间的兼容性问题。

[0052] 隐私保护下的路由协议模拟模块(DO-Simulation)经过转换的输入数据由快速多管理域网络验证InCV的模拟算法DO-Simulation进行处理。DO-Simulation算法模拟了BGP(Border Gateway Protocol，边界网络协议)中的路由通告。然而，在执行DO-Simulation算法期间，参与计算的任何一方(或代理路由器)都不会泄露私有信息。运行DO-Simulation处理之后，网络在收敛后会生成一个秘密数据平面RIBs，这是下一步DO-DPV(DO-Data Plane Validation，数据源平面验证)的基础。

[0053] 隐私保护下的数据平面验证模块(DO-DPV)根据操作员的规范，DO-DPV对不同属性进行验证。对于给定的秘密数据平面，它还保证了所有验证功能的数据无关性。DO-DPV的最终验证结果会公开给所有参与方，以指示在给定配置下，网络中是否满足某些属性

[0054] 快速多管理域网络验证InCV是一个基于云的验证系统，旨在实现网络操作员之间的协作，以确定他们的自治系统(Autonomous System，ASes)和路由器配置是否满足特定属性。通过在计算过程中确保每个AS的配置信息保密，InCV可以防止配置信息在参与者之间泄露。

[0055] 为了更好地说明设计，我们假设每个AS都表现为单个eBGP路由器，通过忽略AS内部的拓扑结构来降低拓扑复杂性。我们认为这个假设是合理的，因为跨域验证主要关注AS之间使用的配置正确性。

[0056] 图5为本公开示例性实施方式的InCV云服务器架构示意图。

[0057] 参考图5,InCV中的每个参与验证的eBGP路由器(例如P0、P1、P3、P4)在云端都会被分配自己的代理服务器,负责执行隐私配置转化并进行安全计算。在计算过程中,每个代理首先将配置转换为SMPC协议的兼容输入,然后与其他代理通信以安全地协同计算输出。由于InCV计算规模庞大,需要数万轮通信和数百GB(千兆字节)的数据流,这种设计是必要的。数据平面之间的信息交互是隐私的,而验证结果则是公开的;网络拓扑也是公开的,可在任一时刻被访问;用户命令也是公开的。

[0058] 基于前述,InCV的云服务架构具有以下优点:更好的性能,通过利用云环境的高带宽和低延迟,InCV的性能可以得到显著提高。较高的隐私性,通过代理端的身份认证及权限控制,InCV确保各个AS的配置信息在计算过程中对其他参与者保密,从而保护了配置隐私。较强的可扩展性,由于代理服务器在云端进行计算,InCV可以轻松地扩展其计算能力以适应不断增长的网络规模。更好的全连接性,通常SMPC计算要求参与方之间的全连接,或至少两两可以互相通信,然而在现实网络中,由于复杂的路由控制,多个AS间的通信很难满足这种要求,而采用云服务架构能够高效解决这一问题。

[0059] 图6为本公开示例性实施方式的配置文件厂商中立化示意图。

[0060] 参考图6,在参与方eBGP路由器与云端的代理服务器交互的过程中,隐私配置转化是一个较为关键的步骤。其目的在于实现配置的厂商中立化。厂商中立(Vendor neutrality)是指一个产品、服务或系统在设计和实现时不偏向于任何特定的厂商或供应商,而能够支持所有厂商的设备输入。在域间网络中,不同AS(例如AS1、AS2和AS3)采用来自不同供应商的路由器和网络设备是非常常见的现象。主流厂商之间的配置语言各不相同,即使实现相同策略也可能会存在巨大差别。

[0061] 为应对这一挑战,我们修改了网络验证工具Batfish的源码,首先将不同厂商的路由器配置转化为相同的中间表示;再将路由器配置文件的中间表示转化为SMPC协议确定的输入格式。通过这种做法解决了不同设备之间的兼容性问题。

[0062] SMPC协议定义了算法实现过程中每个节点的数据结构,每个路由器节点的输入格式均可由域间路由信息表RIB、应用服务器的路由器配置信息Config、应用服务器的路由器节点更新状态HasUpdate和路由信息更新列表UpdateList四部分构成。其中,RIB使用数组存储,数组中的每个条目为一个Route类型的数据结构。Config代表该路由器节点的配置,HasUpdate使用布尔类型数据表示当前节点是否有更新。UpdateList使用数组存储,数组中的每个条目为更新的路由Route及其类型Type,其中Type使用short类型数据用0、1、2分别表示添加、删除、更新三种更新类型。

[0063] Route由Prefix(网络前缀)、NextHop(下一跳)、AsPath(As路径)、LocalPref(本地优先级)、Med(多出口鉴别器)和IDs(路由的ID)这几部分组成。其中,Prefix代表这条路由宣告的前缀,由ip和掩码mask构成。NextHop是一个ip地址,代表下一跳。AsPath使用数组存储,代表需要经过的路由器节点。LocalPref代表这条路由的本地优先级,IDs用于撤销和更新,用路由器名称和序号表示,序号为加入到路由表的时间顺序,保证唯一性;比如A_1。例如A中的路由l1撤销时,邻居B的路由l2对应撤销,B需要判断l2.OriginID＝l1.RIBID来进行删除。

[0064] Config为每个节点的配置,其包含了每个邻居路由器的关系,包括PeerAddress、LocalAs、RemoteAs、Interface和Policy。PeerAddress表示邻居的ip地址,LocalAs表示该

路由器的As号,RemoteAs表示邻居路由器的As号,Interface表示接口的网络协议IP地址。Policy包含入口策略ImportPolicy和出口策略ExportPolicy。ImportPolicy使用数组存储,其中的每个条目为一个路线图routemap,每个routemap包含若干个match语句和set语句,match语句包含Match PrefixList和Match NextHop,用于匹配前缀列表和下一跳;Set语句包含Set LocalPref和Set Metric,它们用于设置本地优先级和Metric。布尔类型的动作用于表明该策略是拒绝deny还是允许permit。

[0065] 图7至14依次为本公开示例性实施方式的DO-Simulation算法1至8的代码示意图。

[0066] 针对隐私保护下的路由协议模拟模块,涉及了八种DO-Simulation算法,下面将结合图7至图14对各个算法进行简要说明。

[0067] DO-Simulation的目标是将多个代理的隐私输入转换为数据平面,这将作为后续步骤中的属性验证的基础。

[0068] 像许多其他验证算法一样,DO-Simulation是针对网络管理员指定的每个ip前缀p执行。同样,我们将宣布ip前缀p的路由器视为公共已知的,它被称为origin(初始)。路由器n接收到的路由通告存储在RIB(n)中,代表路由信息库RIB,它是存储在路由器中的一个数据表,列出了到特定网络目的地的路由,以及与这些路由相关的度量。在算法的初始化步骤,只有RIB(Origin)有一个p的条目,其他路由器的RIB被赋值为空。这是为了建模当origin刚刚发现p时的网络状态。Has_update是一个布尔数组,用于指示路由器是否有要导入的路由通告。类似地,只有has_update(p)会被赋值为True。

[0069] 参考图7,函数Converge()作为算法1,由多次迭代组成,直到网络模拟收敛;也就是说,如果has_update()中的所有元素为假,则网络是收敛的。在每次迭代中,算法选择一个RIB更改过的路由器,并向其对等体发出路由通知。RIB的变化指的是由于插入了一条更优的路由或撤回了最优路由而改变了路由器的最优路由。这些RIB更改存储在update_list中,这是一个具有|n|维度的全局数组,其中|n|是参与方的数量。然后,将RIB更改加载到名为updates(更新)的临时变量中。所有更新都通过ExportRoutes()来建模导出过滤器的动作,将updates转换为exports(输出)。ImportRoutes()函数对对等路由器n上的导入过滤器建模,并类似地修改exports。

[0070] 参考图8,ExportRoutes()算法作为算法2,用于建模导出过滤器,如果更新的路由的类型为add(添加),首先将路由的不传递属性清空;如果AS path包含peer的AS,则不导出该路由。之后,应用export policy来看是否允许导出该路由;如果允许导出该路由,则首先加上as-path发送者的as号并设置下一跳ip。最后将该路由放入exports数组中。如果更新的路由的信息为delete(删除),update(更新)时,则直接导出。

[0071] 参考图9,ImportRoutes()算法作为算法3,用于建模导入过滤器。如果更新的路由的类型为add,且路由的AS path包含peer的AS,则不导入该条路由。对该条路由应用import policy来看是否允许导入该条路由;如果允许导入,则在peer的RIB中添加这条路由。如果更新的路由的类型为delete或update,则在peer的RIB中删除或更新这条路由。

[0072] 参考图10,PolicyOutAllowed算法和PolicyInAllowed算法作为算法4,用于对路由执行导入和导出策略。遍历所有的outPolicy(外部策略),看是否匹配match语句,如果匹配,则执行对应的set语句。

[0073] 参考图11,Match算法作为算法5,用于匹配路由前缀和下一跳,。

[0074] 参考图12,Set算法作为算法6用于设置本地优先级和下一跳。

[0075] 参考图13和图14,UpdateRIB算法作为算法7,用于处理由于导入而导致peer的RIB中的最佳路由发生变化的情况;如果更新的路由的类型为add,则先更新路由的OriginID和RIBID,之后将新的RIB变化添加到update_list(peer),并将has_update(peer)赋值为True,并更新peer的RIB。如果更新的类型为delete和update,则对应路由的撤销和更新,首先匹配OriginID和RIBID是否相等,若匹配,则将新的RIB变化添加update_list(peer),并将has_update(peer)赋值为True,并更新peer的RIB。当路由通告被导入时,它将被插入peer的RIB的第k个位置,在它之前的所有k-1条路由都具有更高的优先级。在对RIB中的路由进行比较的过程是通过PriorityCompare算法(即算法8)来实现的,和BGP中的选路规则一致,首先比较本地优先级,之后比较AS Path的长度和MED。

[0076] 数据无关(data-oblivious)是一种计算方法,它要求在执行计算时,所有操作都不会泄露关于输入数据的任何信息。换句话说,数据无关算法的执行过程不会依赖于输入数据的具体值,因此不能通过观察算法的执行过程来推断输入数据的信息。

[0077] 数据无关算法在隐私保护和安全领域具有重要意义,特别是在安全多方计算(SMPC)和全同态加密等场景中。通过使用数据无关算法,可以在多个参与方之间执行计算,同时确保他们的输入数据不会泄露给其他参与方或第三方观察者。

[0078] 数据无关方法的一个典型例子是基于安全多方计算的排序算法。在这种情况下,多个参与方可以协作对一组数进行排序,但在整个过程中,他们不能获取关于其他参与方输入数据的任何信息。通过这种方式,数据无关方法可以保护参与方的隐私,同时允许他们在不泄露关键信息的情况下进行协同计算。

[0079] 安全外包计算被用来激励这项工作,我们假设在保护的数据上进行计算。这意味着所有的输入和中间结果都不为执行计算的一方或多方所知。除非我们为了访问特定位置的数据而显式地打开它们的值。为了说明的具体性,我们使用符号[x]来指示x的值被保护不受执行计算的实体的影响。

[0080] 为了维护数据隐私,我们必须保证计算方在算法执行期间不会了解任何有关数据的信息。因为假设每个私有值都被充分保护,所以计算方推断关于私有数据的信息的唯一方式是当指令序列或算法的存储器访问模式取决于数据时。因此,为了保证数据隐私,我们正式地将确定性算法的数据遗忘执行公式化如下:

[0081] 定义1:令d表示算法的输入。同样,让A(d)表示算法进行的存储器访问序列。如果对于两个相等长度的输入d和d',算法执行相同的指令序列,并且执行计算的每一方都无法区分访问模式A(d)和A(d'),则该算法被认为是数据无关的。

[0082] 不失一般性,在以下描述中,我们使用算术运算来实现布尔运算。特别地,我们通过a·b来实现合取a∧b,通过(1-a)来实现布尔a的补a。

[0083] 图15为本公开示例性实施方式的数据无关算法的代码示意图。

[0084] DO-Simulation实现了数据无关的执行模式,以防止信息泄漏。在大多数情况下,即使数据被加密,也可以记录访问模式。在这里,我们通过一个在DO-Simulation中高频使用的决定路由优先级的函数priority compare(即图15的策略1和策略2)来说明数据无关算法。根据先前的工作,该算法是数据无关的,因为程序的控制流是固定的并且是公开的,所有各方都可以步调一致地遵循它。此外,对于变量的中间计算,我们采用适合于二进制或

算术电路安全计算的协议来完成。

[0085] 隐私保护下的数据平面验证模块DO-DPV尝试在DO-Simulation生成的秘密数据平面上验证网络属性(例如可达性,隔离等)。作为一个原型,我们假设路由可达性等于转发可达性,并将这些属性转化为对rib的线性扫描操作。例如,为了验证A可以到达B,我们通过对B的RIB执行一个数据无关的搜索,以检查是否有一个到目的地A的有效条目。

[0086] 本公开还进行了路由模拟收敛速度优化,结合了FASTPLANE的路由模拟计算加速技术,通过选择合理的路由传播模拟次序,减少不必要的路由撤销,加快收敛速度,最终提升系统性能。FASTPLANE的关键思想是首先发送全局最优路由公告。对于单调网络,FASTPLANE可以通过选择适当的传播顺序来有效地生成数据平面。然而,FASTPLANE不能应用于非单调网络中。因为在根据上述传播顺序进行模拟时会发生路由撤回,而FASTPLANE不执行撤回操作。为了支持非单调网络,我们实现了基于FASTPLANE的路由撤回操作。在我们所有数据集下的实验表明,这种设计可以通过减少路线撤回的频率来加速模拟过程。然而,这种设计可能会导致特定网络的效率下降,例如每个路由器都喜欢路径更长的路由。

[0087] 我们进行了大量的实验来证明InCV的可行性和好处。

[0088] 图16为本公开示例性实施方式的不同规模网络的时间开销对比图。

[0089] 如图16所示,其横坐标为网络节点的数量(Number of Network Node),纵坐标为验证时间(Verification Time),验证时间的单位为微秒(μs)在SMPC的开销方面,InCV相较于其明文版本用时多了10^7-10^9倍,这也是复杂问题安全多方计算求解程序在性能方面的常见问题。对于32个节点的网络,明文版本的验证器可以在约10us以内完成验证,密文版本的需要约52分钟。

[0090] 图17为本公开示例性实施方式的不同规模网络的通信轮次与全局数量对比图。图17(a)的横坐标为网络节点的数量(Number of Network Node),纵坐标为总的数据传输率(Global Data Sent),总的数据传输率的单位为GB。图17(b)的横坐标为网络节点的数量(Number of Network Node),纵坐标为总的通信轮次(Average Communi-cation Rounds),总的通信轮次单位为k。

[0091] 如图17的(a)和(b)所示,给出了InCV全局通信轮次和通信数据量随网络规模增加的变化趋势。结果表明,在域间验证中实现数据无关的SMPC在时间上是非常昂贵的,因为代理之间必须连续通信数十万次,传输的数据量达到100GB。然而在较小规模的域间网络中,我们认为其绝对耗时依然在可接受范围内。

[0092] 图18为本公开示例性实施方式的InCV与未采用FASTPLANE优化版本的对比图。其横坐标为网络节点的数量(Number of Network Node),纵坐标为验证时间(Verification Time),验证时间的单位为秒(s)。

[0093] 为了展示基于快速控制平面验证FASTPLANE的优化技术的效果,我们在合成网络上比较了优化前后的InCV。如图18所示,结果表明对于具有32个AS的网络,由于路由撤销次数减少和模拟迭代次数减少,实现了约19%的加速。

[0094] 本公开可为多管理域网络中互不信任的多个运营者提供网络验证需求服务,用来在保障域内配置隐私的前提下排查多管理域网络故障,以及确保多管理域网络控制平面的正确性。

[0095] 本公开提出的多管理域网络验证方法及系统,通过结合安全多方计算相关技术,

将网络验证系统的实际适用范围首次从单管理域网络拓展至多管理域网络，并能够保证各参与方数据安全。具体来说，我们设计了模拟BGP中的路由通告的DO-Simulation算法。运行DO-Simulation后，网络在收敛后会生成一个秘密数据平面(RIBs)，这是下一步对不同属性进行验证的DO-DPV算法的基础。

[0096] 本公开提出了基于云服务的多管理域网络验证系统架构，解决了跨域间网络通信高延迟、难以保证数据面可达性等问题，降低了参与者间的通信开销。

[0097] 本公开结合了FASTPLANE的路由模拟计算加速技术，通过选择合理的路由传播模拟次序，减少不必要的路由撤销，加快收敛速度，最终提升系统性能。

[0098] 本公开提供的多管理域网络验证系统，还可以通过采用处理系统的硬件实现方式的装置实现。

[0099] 该装置可以包括执行上述流程图中各个或几个步骤的相应模块。因此，可以由相应模块执行上述流程图中的每个步骤或几个步骤，并且该装置可以包括这些模块中的一个或多个模块。模块可以是专门被配置为执行相应步骤的一个或多个硬件模块、或者由被配置为执行相应步骤的处理器来实现、或者存储在计算机可读介质内用于由处理器来实现、或者通过某种组合来实现。

[0100] 该硬件结构可以利用总线架构来实现。总线架构可以包括任何数量的互连总线和桥接器，这取决于硬件的特定应用和总体设计约束。总线将包括一个或多个处理器、存储器和/或硬件模块的各种电路连接到一起。总线还可以将诸如外围设备、电压调节器、功率管理电路、外部天线等的各种其它电路连接。

[0101] 总线可以是工业标准体系结构(ISA，Industry Standard Architecture)总线、外部设备互连(PCI，Peripheral Component)总线或扩展工业标准体系结构(EISA，Extended Industry Standard Component)总线等。总线可以分为地址总线、数据总线、控制总线等。为便于表示，该图中仅用一条连接线表示，但并不表示仅有一根总线或一种类型的总线。

[0102] 流程图中或在此以其他方式描述的任何过程或方法描述可以被理解为，表示包括一个或更多个用于实现特定逻辑功能或过程的步骤的可执行指令的代码的模块、片段或部分，并且本公开的优选实施方式的范围包括另外的实现，其中可以不按所示出或讨论的顺序，包括根据所涉及的功能按基本同时的方式或按相反的顺序，来执行功能，这应被本公开的实施方式所属技术领域的技术人员所理解。处理器执行上文所描述的各个方法和处理。例如，本公开中的方法实施方式可以被实现为软件程序，其被有形地包含于机器可读介质，例如存储器。在一些实施方式中，软件程序的部分或者全部可以经由存储器和/或通信接口而被载入和/或安装。当软件程序加载到存储器并由处理器执行时，可以执行上文描述的方法中的一个或多个步骤。备选地，在其他实施方式中，处理器可以通过其他任何适当的方式(例如，借助于固件)而被配置为执行上述方法之一。

[0103] 在流程图中表示或在此以其他方式描述的逻辑和/或步骤，可以具体实现在任何可读存储介质中，以供指令执行系统、装置或设备(如基于计算机的系统、包括处理器的系统或其他可以从指令执行系统、装置或设备取指令并执行指令的系统)使用，或结合这些指令执行系统、装置或设备而使用。

[0104] 就本说明书而言，“可读存储介质”可以是任何可以包含、存储、通信、传播或传输程序以供指令执行系统、装置或设备或结合这些指令执行系统、装置或设备而使用的装置。

可读存储介质的更具体的示例(非穷尽性列表)包括以下:具有一个或多个布线的电连接部(电子装置),便携式计算机盘盒(磁装置),随机存取存储器(RAM),只读存储器(ROM),可擦除可编辑只读存储器(EPROM或闪速存储器),光纤装置,以及便携式只读存储器(CDROM)。另外,可读存储介质甚至可以是可在其上打印程序的纸或其他合适的介质,因为可以例如通过对纸或其他介质进行光学扫描,接着进行编辑、解译或必要时以其他合适方式进行处理来以电子方式获得程序,然后将其存储在存储器中。

[0105] 应当理解,本公开的各部分可以用硬件、软件或它们的组合来实现。在上述实施方式中,多个步骤或方法可以用存储在存储器中且由合适的指令执行系统执行的软件来实现。例如,如果用硬件来实现,和在另一实施方式中一样,可用本领域公知的下列技术中的任一项或他们的组合来实现:具有用于对数据信号实现逻辑功能的逻辑门电路的离散逻辑电路,具有合适的组合逻辑门电路的专用集成电路,可编程门阵列(PGA),现场可编程门阵列(FPGA)等。

[0106] 本技术领域的普通技术人员可以理解实现上述实施方式方法的全部或部分步骤是可以通过程序来指令相关的硬件完成,的程序可以存储于一种可读存储介质中,该程序在执行时,包括方法实施方式的步骤之一或其组合。

[0107] 此外,在本公开各个实施方式中的各功能单元可以集成在一个处理模块中,也可以是各个单元单独物理存在,也可以两个或两个以上单元集成在一个模块中。上述集成的模块既可以采用硬件的形式实现,也可以采用软件功能模块的形式实现。集成的模块如果以软件功能模块的形式实现并作为独立的产品销售或使用时,也可以存储在一个可读存储介质中。存储介质可以是只读存储器,磁盘或光盘等。

[0108] 本领域的技术人员应当理解,上述实施方式仅仅是为了清楚地说明本公开,而并非是对本公开的范围进行限定。对于所属领域的技术人员而言,在上述公开的基础上还可以做出其它变化或变型,并且这些变化或变型仍处于本公开的范围内。

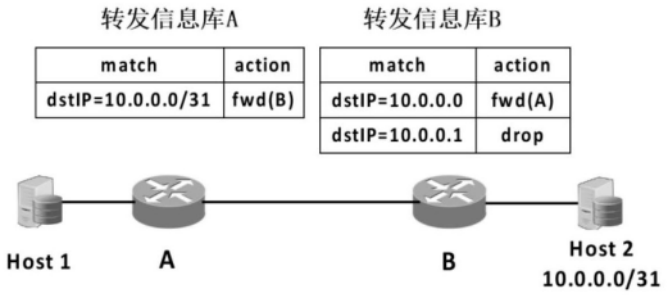

转发信息库A
match
action
dstIP=10.0.0.0/31
fwd(B)
转发信息库B
match
action
dstIP=10.0.0.0
fwd(A)
dstIP=10.0.0.1
drop
Host 1
A
B
Host 2
10.0.0.0/31

图1


路由器B配置文件
access-list 10 deny 10.0.0.0 0.0.0.1
neighbor 65.124.208.93 distribute-list 10 out
65.124.208.93
BGP
Host 1
A
B
Host 2
10.0.0.0/31

图2

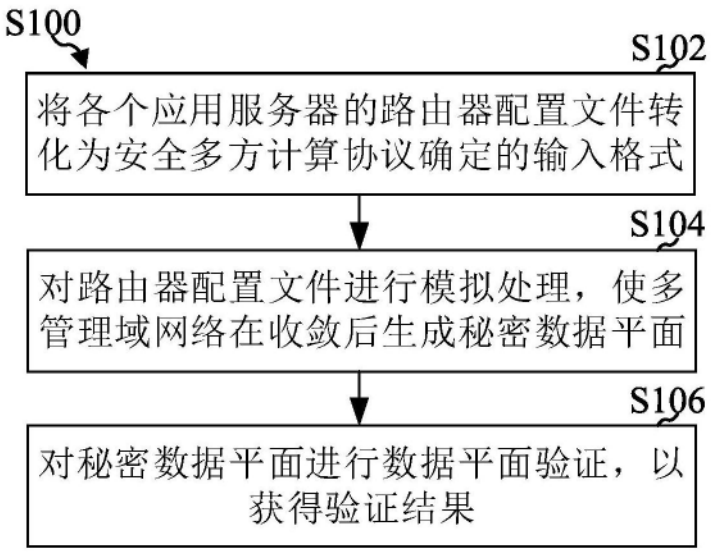

S100
S102
将各个应用服务器的路由器配置文件转化为安全多方计算协议确定的输入格式
S104
对路由器配置文件进行模拟处理，使多管理域网络在收敛后生成秘密数据平面
S106
对秘密数据平面进行数据平面验证，以获得验证结果

图3

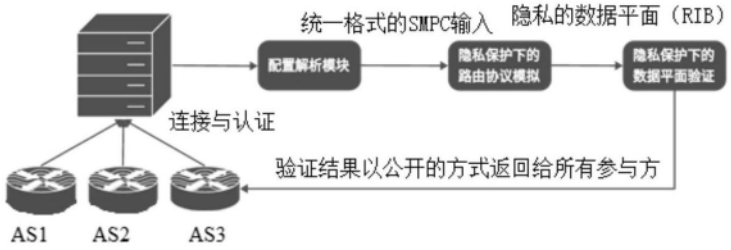

统一格式的SMPC输入
隐私的数据平面（RIB）
配置解析模块
隐私保护下的路由协议模拟
隐私保护下的数据平面验证
连接与认证
验证结果以公开的方式返回给所有参与方
AS1
AS2
AS3

图4

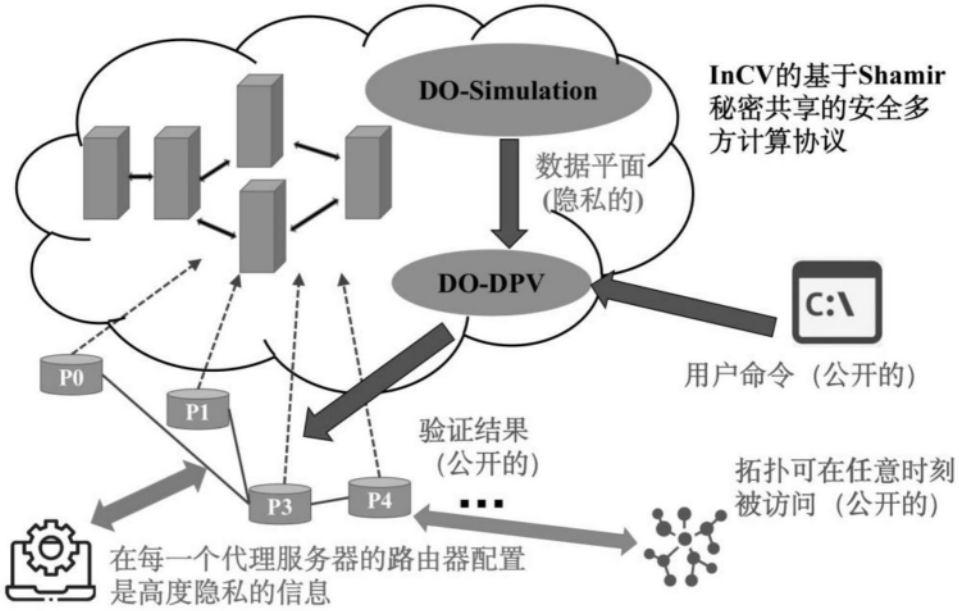

DO-Simulation
InCV的基于Shamir秘密共享的安全多方计算协议
数据平面(隐私的)
DO-DPV
C:\
用户命令（公开的）
P0
P1
P3
P4
验证结果(公开的)
拓扑可在任意时刻被访问（公开的）
在每一个代理服务器的路由器配置是高度隐私的信息

图5

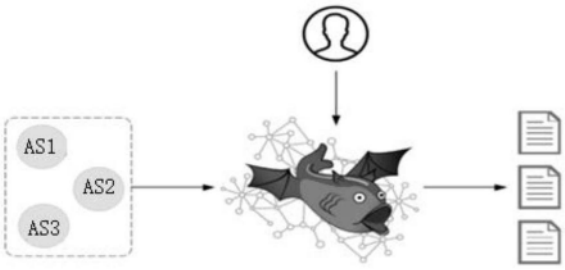

AS1
AS2
AS3

图6

Algorithm 1: DO-Simulation

Input: **Orig** is the announcement node of given prefix, **N** is the set of nodes in the network, OldRIB is the RIB before convergence, C is the configuration changes of each node, and configuration of each node

Output: RIB of each node after convergence

Function `RPVP-Hoyan`:
  // Route initialization
  Init : $n \leftarrow Orig$:
  if $OldRIB(n) = \emptyset$ **then**
    `UpdateRIB(n,`ε`,add)` ;
  end
  Init : $\forall n \in N - Orig$:
    $Rib(n) \leftarrow OldRIB(n), has\_update(n) \leftarrow false, update\_list(n) \leftarrow \perp$;
  // RIB computation
  `Converge();`
end
Function `Converge()`:
  while *true* **do**
    $E \leftarrow \{n \in N \mid has\_update(n)\}$;
    // No update node, converged
    if $E = \emptyset$ **then**
      break ;
    end
    // For efficiency, pick the node closest to Origin
    $n \leftarrow shortest\_pick(E)$;
    // Export and import processes
    for ***each*** $p \in peers$ **do**
      $updates \leftarrow update\_list(n)$;
      $exports \leftarrow$ `ExportRoutes(n,p,updates)` ;
      `ImportRoutes(p,n,exports);`
    end
    // Flag the end of update
    $has\_update(n) \leftarrow false$;
    $update\_list(n) \leftarrow \perp$;
  end
end

图7

Algorithm 2: ExportRoutes

```
Function ExportRoutes(n,p,updates):
    Init: exports ← {} ;
    for each update ∈ updates do
        r ← update.route;
        // Add:  same as the BGP propagation process
        if update.type = add then
            // 1.  transfrom route before export policy is applied.
            // Clear the non-transitive attributes
            r.localpref ← 100 ;
            r.med ← 0 ;
            // do not export if AS path contains the peer's AS
            if peer's AS ∈ r.aspath then
                return false ;
            end
            // 2.  Apply export policy.
            isAllowed ← PolicyOutAllowed(n,p,r);
            if isAllowed = false then
                continue ;
            end
            else
                // 3.  transfrom route after export policy is applied.
                // prepend as-path sender's as-path number
                r.aspath ← r.aspath ∪ n.localas ;
                // Set next-hop ip
                r.nextHoIp ← n.getinterface(p).ip ;
                // 4.  put it into effective exports.
                exports ← exports ∪ (r, add) ;
            end
        end
        // Delete:  directly export
        else if update.type = delete then
            exports ← exports ∪ (r, delete) ;
        end
        // Update:  update the topology condition and export
        else if update.type = update then
            // Same as the step 4 in Add condition
            exports ← exports ∪ (r, update) ;
        end
    end
end
```

图8

Algorithm 3: ImportRoutes

```
Function ImportRoutes(p,n,exports):
    Init: updates ← {} ;
    for each update ∈ exports do
        r ← update.route;
        // Add:  same as the BGP propagation process
        if update.type = add then
            // 1.  transfrom route attribute before import
               policy is applied.
            // skip routes containing peer's AS
            if r.aspath.contains(p.localas) then
                continue;
            end
            // Other atrribute such as Adimin and Protocol are
               ignored.
            // 2.  get the import policy from configuration and
               apply it.
            inPolicy ← p.routemap_n_in;
            // Same as the policy function of export policy
            isAllowed ← PolicyInAllowed(p,n,r);
            if isAllowed = false then
                continue;
            end
            else
                // 3.Update RIB
                UpdateRIB(p,r,add);
            end
        end
        // Delete:  directly update the rib
        if update.type = delete then
            UpdateRIB(p,r,delete);
        end
        // Update:  directly update the rib
        if update.type = update then
            UpdateRIB(p,r,update);
        end
    end
end
```

图9

Algorithm 4: PolicyOutAllowed

```
1  Function PolicyOutAllowed(n,p,r):
       // Get the export policy from configuration and apply it.
2      outPolicy ← n.routemap_p_out;
3      for each subpolicy ∈ outPolicy do
           // Subpolicy coantains multiple match and set
              statements, if all match statments match the route
              successfully, do the behavior of set statements,
              otherwise execute the next subpolicy.
4          matchStmts ← subpolicy.matchStmts ;
5          setStmts ← subpolicy.setStmts ;
6          isMatched ← true;
7          for each match statement m ∈ matchStmts do
               // function match() determine if the attributes of
                  the route conform the statement
8              interface ← n.getinterface(p);
9              if Match(m,r,interface) then
10                 continue;
11             end
12             else
13                 isMatched ← false;
14                 break;
15             end
16         end
17         if isMatched = true then
18             if subpolicy.action = deny then
19                 return false;
20             end
21             else
22                 for each set statement s ∈ setStmts do
23                     Set(s,r)
24                 end
25                 return true ;
26             end
27         end
28     end
       // Default behavior
29     return false ;
30 end
```

图10

Algorithm 5: Match

```
   // Match function for different statement types
1  Function Match(m,r,interface):
2      isMatch ← false ;
3      if m.type = matchPrefixList then
4          for each prefix line line ∈ m.prefixlist do
               // Determine whether the first n bits (mask) are the same
5              if line.prefix.contains(r.prefix.ip) then
6                  if line.action = permit then
7                      isMatch ← true ;
8                  end
9                  break ;
10             end
11         end
12     end
13     else if m.type = matchNextHop then
14         if m.ip = r.ip then
15             isMatch ← true ;
16         end
17     end
18 end
```

图11

Algorithm 6: Set

```
1  Function Set(s,r):
2      if s.type = setLocalpref then
3          r.localpref ← s.localpref;
4      end
5      else if s.type = setMetric then
6          r.metric ← s.metric;
7      end
8  end
```

图12

Algorithm 7: UpdateRIB

```
Function UpdateRIB(p,r,type):
    // routes are stored in a array, a route with a lower index has a
       higher priority
    routes ← RIB(n) ;
    isChanged ← false;
    // Condition 1. Add route into RIB, insert the route into the
       right place of the array based on priority.
    if type = add then
        // Change the id of route, and r.index is the sequence number
           added to the RIB
        r.OriginID ← r.RIBID;
        r.RIBID ← n.name + r.index;
        routes.add(r) ;
        isChanged ← true;
        p.hasupdate ← true;
        p.update_list ← p.update_list ∪ (r, add);
    end
    // Condition 2. Delete route from RIB, delete directly, the
       index needs to be changed accordingly
    else if type = delete then
        for each route e ∈ routes do
            if e.OriginID = r.RIBID then
                routes.delete(e) ;
                isChanged ← true;
                p.hasupdate ← true;
                p.update_list ← p.update_list ∪ (r, delete);
                break;
            end
        end
    end
    // Condition 3: Update the route topology condition. Find the
       route with same id, and update.
    else if type = update then
        for each route e ∈ routes do
            if e.OriginID = r.RIBID then
                isChanged ← true;
                p.hasupdate ← true;
                // When change update_list, delete the update with same
                   id, to avoid duplicate updates
                p.update_list ← p.update_list ∪ (r, update);
                break;
            end
        end
    end
    // If RIB is changed, change the topology condition of less
       preferred routes
    if ischanged = true then
        RIB(n) ← routes ;
    end
end
```

图13

Algorithm 8: PriorityCompare

```
// Compare priority between r1 and r2
Function PriorityCompare(r1,r2):
    // Prefer higher LocalPref
    if r1.localpref ≠ r2.localpref then
        return 1 if r1.localpref > r2.localpref else 0 ;
    end
    // AS path:  prefer shorter
    if r1.aspath.length ≠ r2.aspath.length then
        return 1 if r1.aspath.length < r2.aspath.length else 0 ;
    end
    // Prefer lower Multi-Exit Discriminator (MED)
    if r1.med ≠ r2.med then
        return 1 if r1.med < r2.med else 0 ;
    end
    // By default, prefer the oldest route.
    return 1 if r.RibId < r2.RibId else 0;
end
```

图14

Algorithm 1: PriorityCompare

```
// Compare priority between r1 and r2
Function PriorityCompare(r1,r2):
    // Prefer higher LocalPref
    if r1.localpref ≠ r2.localpref then
        return 1 if r1.localpref > r2.localpref else 0 ;
    end
    // AS path:  prefer shorter
    if r1.aspath.length ≠ r2.aspath.length then
        return 1 if r1.aspath.length < r2.aspath.length else 0 ;
    end
    // Prefer lower Multi-Exit Discriminator (MED)
    if r1.med ≠ r2.med then
        return 1 if r1.med < r2.med else 0 ;
    end
    // By default, prefer the oldest route.
    return 1 if r.RibId < r2.RibId else 0;
end
```

Algorithm 2: Oblivious PriorityCompare

Input: **r1,r2** are two routes with the same origin
Output: a boolean value indicating whether **r2** is more preferred than **r1**

```
Function oblivious-PriorityCompare():
    C0 = r1.lp.equal(r2.lp)
    C1 = r1.len.equal(r2.len)
    C2 = r1.med.equal(r2.med)
    C3 = r1.ID.equal(r2.ID)
    CompareRes =
      (1 − C0) ∗ r.lp.greater_than(r2.lp)
    + C0 ∗ (1 − C1) ∗ r1.len.less_than(r2.len)
    + C0 ∗ C1 ∗ (1 − C2) ∗ r1.med.less_than(r2.med)
    + C0 ∗ C1 ∗ C2 ∗ (1 − C3) ∗ r1.ID.less_than(r2.ID)
    return CompareRes;
end
```

图15

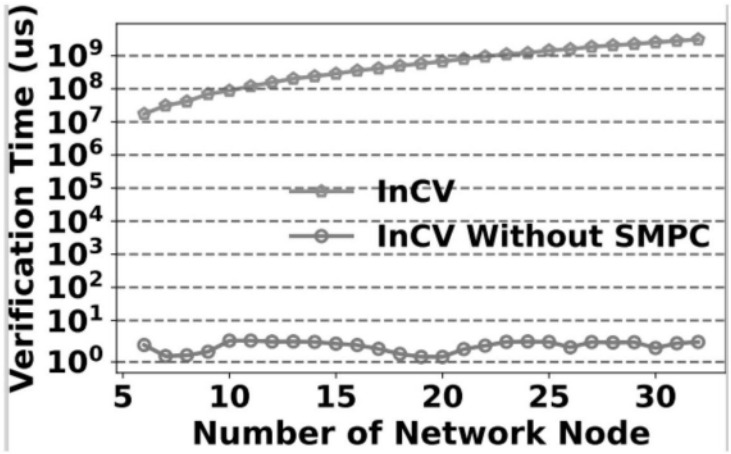
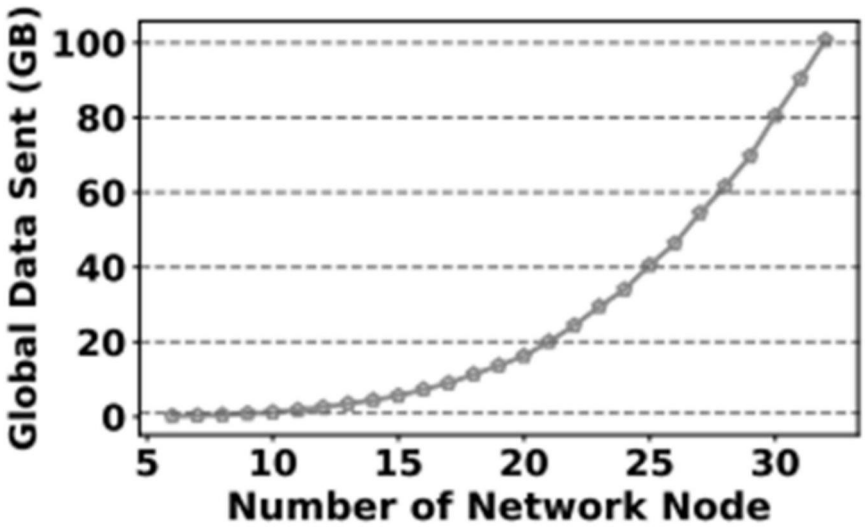
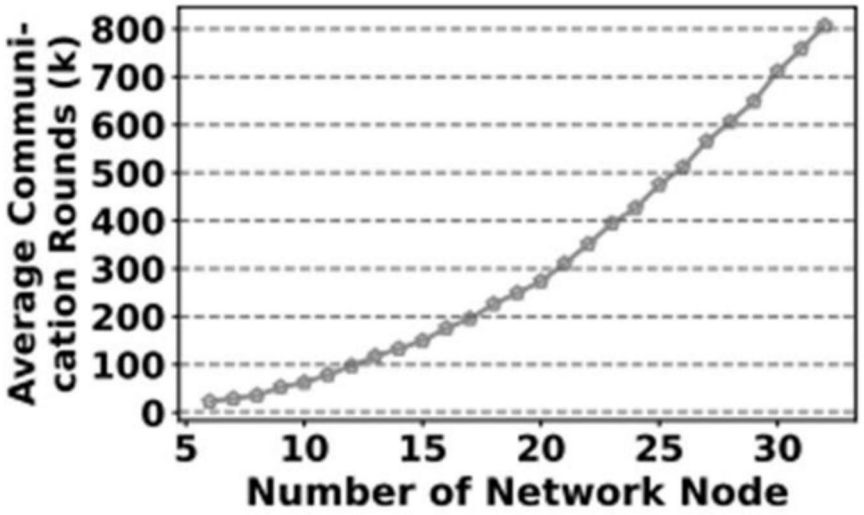


图16


Global Data Sent (GB)
100
80
60
40
20
0
5
10
15
20
25
30
Number of Network Node

（a）


Average Communi-cation Rounds (k)
800
700
600
500
400
300
200
100
0
5
10
15
20
25
30
Number of Network Node

（b）

图17

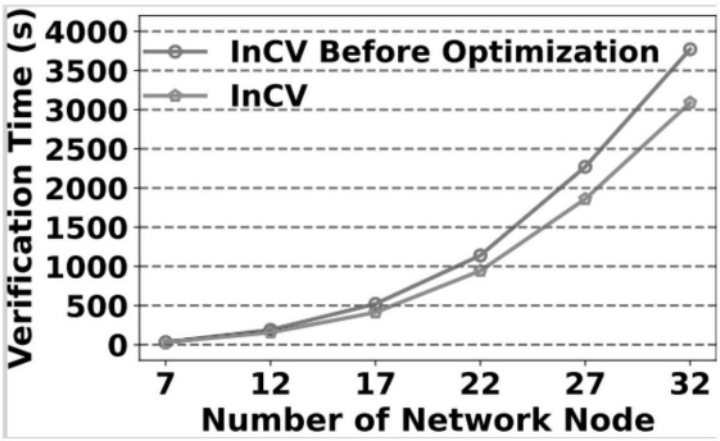

Verification Time (s)
4000
3500
3000
2500
2000
1500
1000
500
0
InCV Before Optimization
InCV
7
12
17
22
27
32
Number of Network Node

图18