面向 HTTPS 的内容分发网络代理关系透明化

王 泽[1,2,3], 李文强[1,2,3], 蔡权伟[1,2*]

[1] 中国科学院信息工程研究所 北京 中国 100093
[2] 中国科学院数据与通信保护研究教育中心 北京 中国 100093
[3] 中国科学院大学网络空间安全学院 北京 中国 100049

摘要 内容分发网络可以提高浏览器访问网站的速度和网站安全性(例如防 DDoS 攻击), 已被广泛部署和应用。当浏览器的 HTTPS 连接被重定向到内容分发网络的代理服务器时, 由于浏览器要求收到的证书与访问的网站域名匹配, 内容分发网络不能直接使用自己的证书, 而是需要使用内容来源网站的有效证书和私钥, 才能正确地同浏览器建立 HTTPS 连接。现在, 大量内容分发网络和源网站共用私钥, 违背了 PKI 的设计原则。同时, 现有模式下的代理关系对于浏览器是不透明的, 内容源网站也无法快速地撤销代理关系。本文提出了一种面向 HTTPS 的, 公开可验证的内容分发网络代理关系管理方案。在本方案中, 内容来源网站在公开的日志服务器中发布授权代理其内容的内容分发网络的信息, 并产生可验证的代理证据。浏览器使用 HTTPS 连接内容分发网络代理服务器时, 代理服务器推送自己的证书和相应的代理证据。浏览器不必与源网站直接通信, 即可验证代理证据, 并使用内容分发网络的证书正确地建立 HTTPS 连接。同时, 本方案允许内容分发网络和内容来源网站独立地管理各自的私钥, 且内容来源网站可以独立地更新、撤销代理关系。我们实现了本方案的原型系统, 并进行了性能评估: 实验结果表明, 本方案在带宽/延迟/存储等各方面均未造成过大开销。

关键词 公钥基础设施; 内容分发网络; 安全套接层; 安全传输层协议; 公开日志; 代理; 信任
中图法分类号 TN393.08 **DOI 号** 10.19363/j.cnki.cn10-1380/tn.2018.03.02

Delegation Transparency for HTTPS with CDNs

WANG Ze[1,2,3], LI Wenqiang[1,2,3], CAI Quanwei[1,2*]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2] Data Assurance and Communications Security Center, Chinese Academy of Sciences, Beijing 100093, China
[3] School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract CDNs (Content Delivery Networks) have been widely deployed to achieve fast content access and better security such as DDoS attack mitigation. However, to support HTTPS services, CDN providers need their custom original websites to share certificates and private keys to establish HTTPS connections with browsers, because browsers require validating the visited website's certificate rather than the connected CDN's certificate. Sharing private keys explicitly conflicts the design principle of PKI and arises deep concerns over CDN's role in standard PKI trust model, considering that the delegation between the CDNs and the original websites is opaque and the original websites lack the capability of rapid updating and revoking such delegation. We present DET (Delegation Transparency for HTTPS with CDNs), a system that provides public and verifiable delegation management for HTTPS with CDNs. In DET, an original website publishes all its delegated CDN providers in a public log, and generates a corresponding delegation proof, which is delivered to the visiting browser along with the CDN's certificate during HTTPS establishment. The visiting browser is able to verify the proof and accept the CDN's certificate without any previous connections to the original website. In DET, the original websites and CDNs manage their SSL private key separately, meanwhile the original websites are able to revoke or update their delegation independently. We have implemented the prototype of DET. The performance evaluation demonstrates that the introduced overhead (in terms of bandwidth, latency and storage) is modest.

Key words public key infrastructure, content delivery network, SSL, TLS, public log, delegation, trust

通讯作者: 蔡权伟, 博士, 助理研究员, Email: caiquanwei@iie.ac.cn。
本课题得到 973 课题《租户可控的云数据安全理论与方法研究》资助。
收稿日期: 2017-10-17; 修改日期: 2017-12-07; 定稿日期: 2018-02-05

1 引言

内容分发网络(Content Delivery Network, CDN)是当前广泛应用的网络基础服务, 可以为其客户(内容的来源网站, 以下简称源网站)提供快速、可伸缩的用户访问和更好的安全保障。CDN 在广泛的地理范围内部署大量的代理服务器, 并在代理服务器中缓存源网站的内容供浏览器快速访问。由此, CDN 缩短了浏览器到服务器的传输距离, 并分散了源网站服务器的流量压力。同时, CDN 也提供多种安全增强服务, 如应用层防火墙、HTTPDNS 和 DDoS 防范。研究[1-3]表明, 近年来 CDN 市场规模的年均增速在40%以上。到 2019 年, 预计超过 50%的网络流量将通过 CDN 加速。

当浏览器通过 HTTPS 访问源网站时, 访问的连接可能会被重定向到附近的 CDN 代理服务器, 而不是源网站的服务器。浏览器会验证收到的证书是否和访问的源网站的域名匹配。此时, 如果代理服务器直接应用 CDN 自身的证书, 会使浏览器验证证书失败。CDN 只有提供源网站的证书并使用对应的私钥才能同浏览器成功地建立 HTTPS 连接。

因此, 目前主流 CDN 以和源网站共用证书私钥的方式提供 HTTPS 服务。现有的共用私钥的方式主要有两种[4,5]: 1. CDN 直接使用“客户证书”, 即由源网站向 CDN 提供有效证书和对应的私钥; 2. CDN 使用“共享证书”, 此时 CDN 向自己的 CA 申请证书, 并将源网站的域名添加到主体可替换名称(Subject Alternative Name)证书扩展中。在这两种方法中, CDN 都拥有了源网站的有效证书和对应的私钥。共用私钥的方案存在以下严重问题:

1) 源网站私钥易泄露。CDN 拥有大量源网站的私钥, 并需要将源网站私钥部署在大量代理服务器中。一旦其中一个代理服务器被攻破, 该代理服务器中存储的源网站私钥均会泄露。这导致 CDN 难以在安全性要求高的网站中(如银行)进行应用。

2) 源网站无法独立、即时地撤销代理关系。目前代理关系是由共用私钥体现的, 撤销代理关系需要源网站向自己的或者 CDN 的 CA 申请撤销相应的证书。由于源网站多使用较为昂贵的企业型证书(Organization Validation SSL Certificate, OV)或增强型证书(Extended Validation SSL Certificate, EV), 证书撤销和重新签发会引入额外的经济开销, 并导致代理关系撤销不及时[2]。

3) 代理关系对浏览器不透明。访问源网站的浏览器无法确定内容的来源是 CDN 代理服务器还是源网站的服务器, 也无法验证代理关系的有效性。本质上, 这是因为传统的 SSL/TLS 协议将端对端的身份认证和内容的机密性、完整性保护紧密耦合; 而 HTTPS 连接中 CDN 仅仅是源网站内容的镜像, 并非身份的镜像, 二者存在着矛盾。

而且, 随着 CDN 级联[6]技术方案的应用, 上级 CDN 进一步地将源网站的内容代理至下级 CDN, 上述的关于密钥管理、代理关系管理和代理关系验证的问题会变得更加严重。例如, 当源网站撤销上级 CDN 向下级 CDN 的内容代理关系时, 需要考虑多级 CDN 的关系。

针对这些问题, 我们提出了一种发布、管理、验证源网站和 CDN 之间内容代理关系的方案 DET (Delegation Transparency for HTTPS with CDNs)。DET 设有日志服务器, 公开地记录所有源网站和 CDN 之间的代理关系。日志通过公开的、唯一的、前后一致的、基于树的结构, 展示所有参与的源网站和 CDN 的代理关系信息, 从而实现源网站代理关系的公开可验证和查询。同时, 日志服务器也周期性地收集源网站和 CDN 对日志信息的操作, 并将执行过的操作添加在公开的操作记录中, 作为审查的依据。

与传统的证书透明化方案不同, DET 将源网站和 CDN 的代理关系(而不是证书)记录在公开的日志中。我们使用有向无环图描述源网站到 CDN 的代理关系, 使得源网站可以使用哈希值表示完整的代理关系, 且代理关系可以被高效地传输和验证。在代理关系的计算时, 我们引入了变色龙哈希函数, 使得 CDN 的 SSL/TLS 密钥改变不影响代理关系的表示, 此时源网站无需更新代理关系。

源网站为 CDN 产生基于密码学的证据, 证明 CDN 被授权代理其内容, 以及(可选的)CDN 级联时内容从源网站到该 CDN 的分发路径。浏览器结合公开日志中的代理关系信息验证上述证据。

日志服务器可以被任何利益相关方(例如源网站、CDN 或公共服务机构等)随时、公开地监督, 检查其中发布的信息是否正确。一旦这些信息发生错误, 利益相关方可以及时发现。同时, 日志服务器可以被公开审查。日志服务器产生公开的基于密码学的证据, 使得其无法在不留下证据的情况下作恶。网络中的任何实体, 均可充当审查机构, 根据日志服务器提供的证据在线地验证日志服务器的状态, 确保日志服务器的正确运行。

通过将内容分发网络的代理关系透明化, DET 提供了如下现有方案不具备的、对 CDN 应用非常重要的特性。

1) 独立的密钥管理

DET 将内容代理的授权和共用私钥解耦合。在 DET 中, 内容代理关系通过日志服务器中的日志信息被显式地表达。CDN 和源网站独立拥有、管理各自的 SSL/TLS 公私钥。SSL/TLS 公私钥的更新不会影响源网站和 CDN 之间代理关系的表达。这避免了 CDN 集中管理大量源网站的私钥, 或使用一个绑定了大量源网站域名的私钥。

2) 主体可控的代理关系

源网站可以独立、实时地在日志服务器中管理(初始化、更新、撤销)自身的所有代理关系, 不需要借助自己或者 CDN 的 CA。源网站可管理的代理关系, 包括从自身到CDN的内容代理, 也包括(CDN级联时)上级 CDN 到下级 CDN 的内容代理。

3) 浏览器可知

当浏览器通过 HTTPS 连接 CDN 时, CDN 代理服务器在 SSL/TLS 连接握手消息中加入源网站提供的证据; 浏览器验证通过后, 使用 CDN 自身的证书建立 HTTPS 连接。支持 DET 的浏览器可根据收到的证据明确得知接收到的内容来自源网站还是 CDN; 在 CDN 级联的情形下, 浏览器也可验证从源网站到被访问 CDN 间的多级代理关系。

4) 高效操作

日志服务器将各个源网站和CDN的信息记录在二叉搜索树中, 为依赖方提供高效的查询服务。日志信息的审查、监督和验证仅包含哈希值计算和签名验证操作, 避免了对浏览器、源网站和审查机构造成过高的性能开销。

我们实现了 DET 原型系统, 包括 DET 日志服务器和支持 DET 中 HTTPS 连接方式的 CDN 代理服务器和浏览器。在 HTTPS 连接建立过程中, 浏览器验证 CDN 的证书和代理证据。我们也实现了源网站和 CDN 的管理工具, 帮助它们管理和监督在日志中的信息。性能测试表明, 我们的方案中浏览器的带宽、计算等各方面开销和日志服务器各项操作效率都是可接受的。

本文的后续章节安排如下。第 2 章介绍了本方案的系统模型和安全假设; 第 3 章说明了公开日志的结构; 第 4 章详细阐述了各方的交互过程; 第 5 章和第 6 章对方案的安全性和性能进行了分析; 第 7 章讨论了技术方案的扩展; 第 8 章介绍了相关的工作; 第 9 章是全文总结。

2 系统模型与安全目标

DET 的设计目标, 是提供可显式声明、独立管理、即时撤销、高效验证源网站和 CDN 之间的内容代理关系的公开日志系统。

2.1 系统角色

DET 的系统模型包含五个主要角色: 日志服务器、源网站、CDN、浏览器和审查机构。我们将在对应的小节中详细介绍每个角色在系统中的功能和相关的安全假设。为了使 DET 方案与现有的 PKI、SSL/TLS 和 CDN 应用兼容良好, 我们考虑 DET 运行在现实的网络情况下, 即: CDN 和源网站都配置了有效的 PKI 证书; 浏览器和日志服务器可以正确地验证数字证书; 各角色间有较为宽松的时间同步(允许数分钟级别的误差)。

我们假设使用的密码学方法是安全的。在原型系统中, 我们使用了 SHA256 作为通用的哈希函数, 使用了基于离散对数的变色龙哈希函数, 以及 RSA-2048 签名/验签算法。

2.1.1 日志服务器

DET 将源网站和 CDN 之间的代理关系记录在日志服务器中。日志服务器的内容是公开可见的, 为方案各依赖方提供一致的信息。我们仅假设日志服务器可以保证一定程度的可用性, 方案各方不必信任日志服务器。日志服务器可以向审查机构提供证据, 证明其按照预定的规则运行。日志服务器也能为源网站或 CDN 产生证据, 证明相应的信息确实存在于日志中。

日志服务器拥有属于自己的私钥, 为上述证据签名。与证书透明化方案[7]类似, 日志服务器的公钥可以预先部署在方案各方之中, 也可以从日志服务器的有效数字证书获取。同时, 日志服务器维护一个自己的信任根证书列表, 包括了所有被主流浏览器和操作系统信任的根证书。

2.1.2 CDN

CDN 将源网站的内容缓存在代理服务器中, 并将内容分发至来访的浏览器。CDN 可能从源网站直接获得其内容, 也可能从另一个 CDN 处间接获得(称为 CDN 级联)。如果是后者, 获取内容之前 CDN 需要确认另一个 CDN 从级联关系上是否确实是自身的上游(详见 7.2 节)。

本方案中 CDN 使用自己的证书公钥与浏览器建立 HTTPS 链接, 不需要与源网站共用证书和私钥。CDN 将自身的信息提交并记录在日志服务器中。CDN 可以随时访问日志服务器, 确认关于自身的信息正确。

2.1.3 源网站

为了提供更好的用户体验, 源网站授权 CDN 代理并分发自己产生的内容。源网站独立地管理自己

的内容代理关系, 包括哪些 CDN 被授权代理自身的内容和内容分发的具体路径。相应地, 源网站为 CDN 生成授权的凭据。源网站将自身的信息和代理关系提交、记录在日志服务器中。源网站可以随时监督日志服务器, 确认相关的信息正确。

源网站拥有有效的数字证书。日志服务器和浏览器可以通过验证数字证书得到源网站的公钥。源网站不需要和 CDN 共用证书和私钥。

2.1.4　浏览器

浏览器通过 HTTPS 连接访问源网站。访问时, 浏览器的连接可能被定向到源网站的服务器, 也可能被定向到 CDN 的代理服务器。在两种情况下, 浏览器都能成功地建立 HTTPS 连接, 并可以分辨这两种情况。当连接到 CDN 代理服务器时, 浏览器能验证该 CDN 是否被源网站授权代理其内容。此时, 浏览器在使用标准方式验证证书之外, 也能够验证 DET 方案产生的证明 CDN 代理关系的证据。验证通过后, 浏览器使用 CDN 的公钥建立 SSL/TLS 连接。

2.1.5　审查机构

在 DET 中, 审查机构的主要功能是检查日志服务器是否按照预先设定的规则正确地运行。审查的方式是验证日志服务器提供的证据。由于所有的证据均可由日志服务器在线提供, 审查可以在线独立完成, 所以审查机构可以由网络中的任何一方担任。因此, 网络中可以存在大量冗余的、互为备份的审查机构。例如, 每个 CDN 代理服务器都可成为一个审查机构。每个审查的地位是平等的, 不存在“单点失效”的风险, 即: 若某些审查机构被敌手控制, 其他审查机构的功能不受影响。我们假设对于任何浏览器、源网站、CDN, 都有较多的审查机构可供选择和交互。此外, 审查机构可以帮助浏览器等验证 CDN 提供的关于代理关系的证据, 并返回结果。

综上, DET 中各角色的相互关系如图 1 所示。

2.2　安全目标

DET 方案的目标是, 在源网站和 CDN 不共用私钥的前提下, 浏览器可以和源网站授权代理内容的 CDN 正确地建立 HTTPS 连接。浏览器不会和未被授权的 CDN 代理服务器建立 HTTPS 连接。

源网站对 CDN 的内容代理关系是源网站产生相关证据、浏览器验证证据的依据。代理关系记录在日志服务器中。日志的内容是否真实可靠, 是安全目标能否成功达成的关键。由于日志服务器是公开的, 其中的所有信息均可在线地获取。源网站和 CDN 应定时确认日志中自身信息正确。为了防止日志服务器向不同的浏览器/源网站/CDN 提供不同的关于特定代理关系的描述, 我们要求日志向各方提供相同的日志内容(称为“唯一性”)。

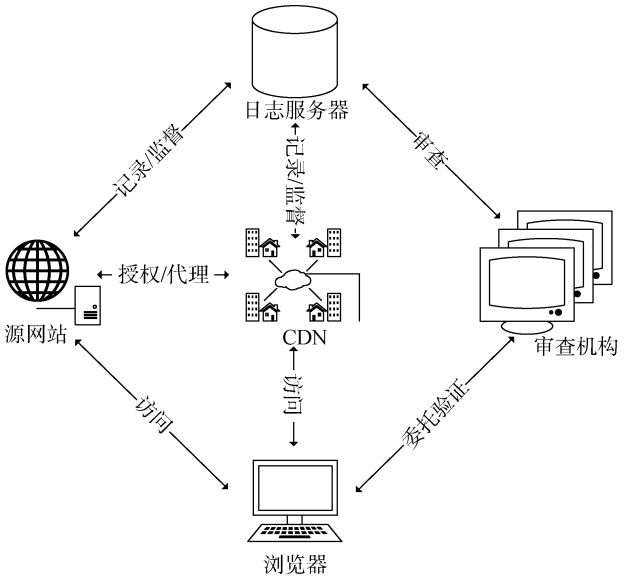

图 1　DET 中系统各角色和相互关系的示意图

Figure 1　System model of DET

审查机构检查日志的操作是否符合预先设定的规则、日志服务器的最终状态和进行过的操作是否一致。为了防止日志服务器篡改已经经过审查的历史记录, 保证日志服务器提供的操作记录是前后一致的、未被篡改的(称为“一致性”), 我们要求要求日志服务器的操作历史是仅可添加的(Append-only), 即: 日志服务器不能回退或否认进行过操作; 任何操作, 都会留下公开、永久的证据, 并在之后的任何时刻均可被检查。

我们为日志服务器设计了可以公开获取和验证的、基于密码学的证据, 证明日志服务器满足以上性质(详见 3.2、3.3 节)。

3　公开日志的结构

本章中, 我们首先在 3.1 节介绍源网站到 CDN 的内容代理关系的表示, 以及代理摘要的生成。代理摘要和其他相关信息, 会记录在公开日志中。

在 DET 中, 日志服务器的日志由两部分组成。一部分称为“状态树”, 另一部分称为“操作记录”。状态树是二叉搜索树, 记录了每个源网站的代理关系和 CDN 的特定公钥的最终状态。操作记录维护了仅可添加的日志操作的完整历史。操作记录由一系列 Merkle Hash 树时序地级联而成。日志的两部分相互配合, 为各类日志功能提供高效的、基于密码学的证据。我们在 3.2 节和 3.3 节依次介绍状态树和操作记录的具体结构。

3.1 代理拓扑结构和代理摘要

代理拓扑结构。源网站到 CDN 的内容代理关系可以由相应的代理拓扑结构表示。代理拓扑结构完整地描述: 1)源网站授权代理内容的所有 CDN; 2)每个 CDN 获得源网站内容的具体途径。为了完整地支持 CDN 应用, 代理拓扑结构可以表示 CDN 级联, 即源网站将内容代理到上游 CDN, 上游 CDN 进一步将内容代理到下游 CDN。

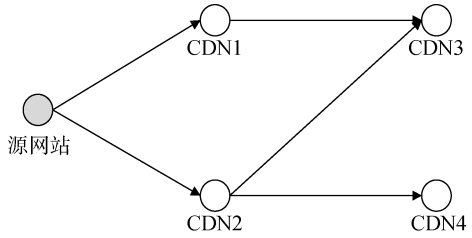

图 2 代理拓扑结构示意图

Figure 2 The structure of delegation topology

源网站的代理拓扑结构被抽象为一个有向无环图(Directed Acyclic Graph, DAG)。如图 2 所示。图中有唯一源节点, 代表源网站; 每个非源节点代表不同的 CDN; 图中任何一条有向边, 表示由上游内容提供者(源网站或者 CDN)向下游 CDN 的代理关系。

在代理拓扑结构中, 每个 CDN 节点表示为:

CDN 节点 :=<CDN 域名, SSL/TLS 公钥>

其中 *SSL/TLS 公钥*即为浏览器和 CDN 代理服务器建立 HTTPS 连接时 CDN 使用的公钥。当浏览器和 CDN 建立 HTTPS 连接时, CDN 应提供有效的数字证书, 且证书的主体名和公钥应和 CDN 节点的内容匹配。

在代理拓扑结构中, 源网站的节点表示为:

源网站节点 :=<源网站域名>

代理摘要。代理摘要是根据代理结构生成的一个哈希值。计算代理摘要的方式类似于计算 Merkle 哈希树的根节点。我们对代理拓扑结构的每个节点计算哈希值, 并将源网站节点的哈希值称为代理摘要。如果源网站没有使用 CDN, 则代理摘要记为 NULL。

源网站节点的哈希值为:

源网站节点的哈希 :=

Hash{源网站域名||所有子节点的哈希}

其中源网站节点的子节点, 指从源网站节点出发的有向边的终点。所有子节点的哈希按照子节点的域名的字典顺序排列。需要注意的是, 源网站节点的哈希并不是简单地对源网站节点的内容计算散列值, 而是应包含所有子节点的信息。

进一步地, 对于 CDN 节点, 节点的哈希值的计算方式为:

CDN 节点的哈希 :=

Hash{Hash{域名||CH{公钥, r}}||所有子节点的哈希}

其中, *Hash* 代表一般的哈希函数(如 SHA256); *CH* 代表变色龙哈希函数(Chameleon Hash, CH), *r* 是一个随机数。变色龙哈希函数[8]是一类特殊的设有秘密陷门的哈希函数。拥有陷门的一方容易计算出哈希的碰撞; 没有陷门的一方则很难计算出碰撞(与普通哈希函数具有相同性质)。

CDN 节点的哈希的计算过程中所使用的变色龙哈希函数的陷门, 由节点对应的 CDN 拥有。每个 CDN 都拥有唯一的陷门。计算变色龙哈希函数之前, 需要使用陷门对应的公钥初始化变色龙哈希函数。本方案中, CDN 将自己的变色龙哈希函数公钥(下文简称为变色龙公钥)公开发布在日志服务器中。

由于使用了变色龙哈希函数, 当 CDN 由于证书更新、私钥泄露或其他原因改变其 SSL/TLS 公钥时, 可以为新公钥找到碰撞(即新的随机数 *r*)。在新的公钥和相应随机数下, 代理拓扑结构中该 CDN 节点的哈希可以保持不变, 从而代理摘要保持不变。因此, CDN 可以独立地管理自身的 SSL/TLS 公钥, 不引起源网站额外的操作。

直接/多步代理证据。当连接到 CDN 代理服务器时, 浏览器应要求 CDN 提供证据, 证明 CDN 被授权代理源网站的内容。证据分为直接代理证据和多步代理证据。其中, 直接代理证据使得浏览器能够直接验证相应 CDN 是否具有目标源网站的代理授权; 多步代理证据不仅证明了代理授权, 还能够体现代理路径中的任意一步或几步。

若浏览器已知源网站的域名和代理摘要, 以及 CDN 的变色龙公钥, 直接代理证据使得浏览器可以根据 CDN 的域名和 SSL/TLS 公钥计算出代理摘要, 进而使得浏览器能够验证对 CDN 的代理授权。

在代理拓扑结构中, CDN 的直接代理证据由两部分组成: 1)CDN 域名和 SSL/TLS 公钥(即 CDN 节点的内容), 和随机数 *r*(由对应的 CDN 给出), 2)计算代理摘要所必需且最少的额外的哈希值(由源网站预先提供给 CDN)。例如, 在图 2 中, *CDN1* 的直接代理证据, 由 *CDN3 节点的哈希*, *CDN1 节点的内容和随机数* r_1, *CDN2 节点的哈希*组成。

多步代理证据可以由直接代理证据扩展得到: 只需要将直接代理证据中多步代理路径上的其他 CDN 节点的哈希, 相应地替代为这些 CDN 节点的内容、计算变色龙哈希的随机数和它们子节点的哈希即可。例如, 在图 2 中, 证明源节点到 CDN3 的多步

代理证据, 由 *CDN3 节点的内容和随机数 r_3*, *CDN1 节点的内容和随机数 r_1*, *CDN2 节点的哈希*组成。

3.2　状态树

状态树是二叉搜索树, 记录源网站的代理摘要和 CDN 的变色龙公钥。状态树的每一个节点均代表一个源网站或者 CDN; 每一个源网站或 CDN 在状态树中仅有一个节点。节点按照源网站和 CDN 域名的字典顺序进行排序。一个状态树的示例如图 3 所示。

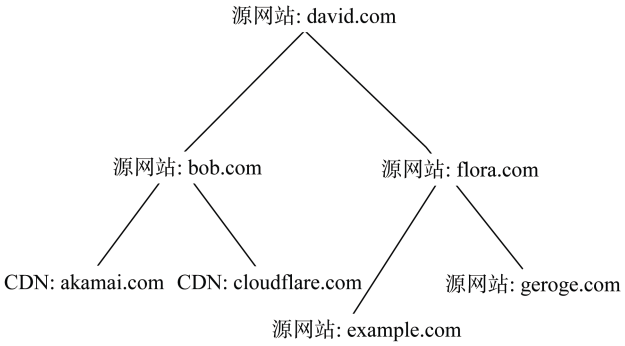

图 3　日志的状态树示意图

Figure 3　The structure of state tree of public log

状态树中代表源网站的节点表示为:

源网站节点 :=<源网站域名, 代理摘要>

状态树中代表 CDN 的节点表示为:

CDN 节点 :=<CDN 域名, 变色龙公钥>

由于每个源网站和 CDN 在状态树中均只对应一个节点, 因此源网站的代理摘要和 CDN 的变色龙公钥都是唯一的。

状态树的哈希。状态树的哈希是树根节点的哈希值。与计算代理摘要类似, 状态树中每个节点的哈希计算为:

节点的哈希 :=

Hash{Hash{域名||代理摘要或变色龙公钥}||左子节点的哈希||右子节点的哈希}

节点的存在性/不存在性证据。如上节所述, 浏览器验证直接代理证据时, 需要从状态树中获取源网站的代理摘要和 CDN 的变色龙公钥。浏览器从状态树获取这些信息时, 状态树应能相应地证明这些信息在树中确实存在。此外, 源网站和 CDN 监督日志服务器, 检查日志中自身的信息是否正确时, 也要求状态树证明提供的信息确实在日志中存在。所以, 状态树应能够对树中的节点提供存在性证据。

若已知状态树的哈希, 状态树中节点的存在性证据由两部分构成: 1)节点的内容, 2)计算出状态树的哈希必需且最少的所有额外的哈希值。例如, 在图 3 中, *bob.com* 的存在性证据, 由 *akamai.com 节点的哈希*, *cloudflare.com 节点的哈希*, *bob.com 的域名和代理摘要*, *flora.com 节点的哈希*, *Hash{david.com||david.com 的代理摘要}*组成。

有的时候, 状态树也需要给出某个源网站或 CDN 的不存在性证据。例如, 未加入方案的网站需要确定在日志服务器中不存在自己的记录, 或者从方案中注销的网站需要确定日志服务器已经在状态树中删除了自己的节点。

不存在性证据由两个相邻节点的存在性证据组成。例如, 对于节点 *X*, 其不存在性证据是节点 *A* 和 *B* 的存在性证据, 其中 *A*<*X*<*B*, 且 *A* 是 *B* 的左子树的最右节点或者 *B* 是 *A* 的右子树的最左节点。由此可以断定状态树中不存在 *X*。

由于采取了二叉搜索树结构, 在状态树中搜索源网站或 CDN, 以及提供存在性/不存在性证据的时间和空间复杂度为 *O(logN)*, *N* 为状态树中源网站和 CDN 的数量。

3.3　操作记录

操作记录是仅可添加的数据结构, 时序地记录源网站和 CDN 对代理摘要或变色龙公钥的所有操作。日志服务器周期性地批处理收到的操作, 更新状态树, 并将执行过的操作添加到操作记录中。

在每个更新周期, 操作记录将所有源网站或 CDN 的操作按照出现的时间顺序排列并组成一个 Merkle 哈希树, 作为当前更新周期的操作树。操作树的叶子节点为源网站或 CDN 的操作的消息。日志服务器支持的具体操作类型和消息格式将在下一章介绍。

当前更新周期的操作树也包含上个更新周期的操作树树根和更新后的状态树的哈希。最后, 每个更新周期的操作树时序地级联, 组成完整的操作记录。图 4 是操作记录的示意图。

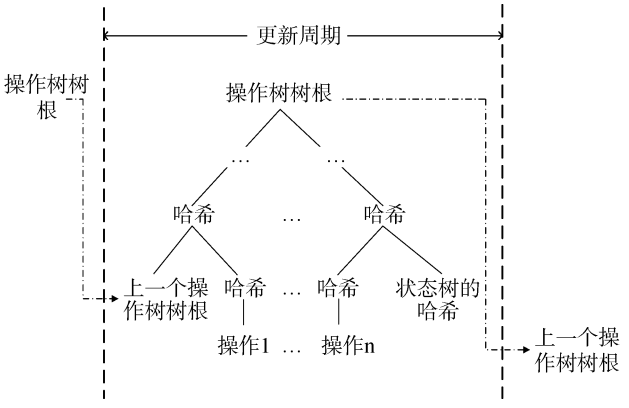

图 4　公开日志的操作记录示意图

Figure 4　The structure of operation record of public log

唯一性证据。前文提到, 日志服务器应该向各依赖方提供同样的日志内容。由于哈希函数的单向性, 如果每个更新周期的操作树以及状态树的哈希是唯一的, 则可认定日志是唯一的。因此日志每次更新后, 应签名发布最新的操作树树根、状态树的哈希、此次更新的时间和下次更新的时间, 称为签名的树根。审查机构会比较各种渠道得到的签名的树根。如果审查机构发现同一时刻有多个不同的签名的树根, 则向公众公布这些的树根, 作为日志服务器违反唯一性的证据。其他依赖方根据自身安全设置做出应对。例如, 浏览器不再根据该日志服务器的记录验证代理证据。

一致性证据。一致性证据可以证明操作记录的仅可添加属性, 即操作记录仅添加新的操作, 而不会删除或篡改已经记录在日志中的操作。对于相邻的两个更新周期, 一致性证据由相邻周期签名的树根, 以及前一周期操作树树根到后一周期操作树树根的 Merkle 哈希树认证路径组成。任意两个更新周期间的一致性证据可以由上述方式相应地扩展得到。由于哈希函数的单向性, 一致性证据包含了旧日志的完整历史, 保证了新的日志是仅仅在旧日志基础上增加某些操作产生的, 没有对旧日志做出任何更改。

正确性的检查。日志服务器正确性的检查, 以及状态树和操作记录是否匹配的检查, 需要审查机构下载日志的完整内容。正确性的检查需要下载和执行一个历史时期内日志的所有操作, 因此开销较大。但是审查行为是独立于浏览器和 CDN 代理服务器建立 HTTPS 这一过程的, 因此方案对正确性的检查的实时性要求不高。此外, 多个审查机构可以分工合作, 每人仅检查其中若干个更新周期中日志的正确性, 从而降低审查行为的开销。

4 系统操作

4.1 日志服务器支持的操作

日志服务器支持源网站和 CDN 进行表 1 中包含的所有操作。所有提交的操作都应附带源网站或 CDN 的有效证书。有效的证书应由日志服务器信任的 CA 签发。操作的消息应由证书对应的私钥签名, 以证明提交的操作确实由相应的源网站或 CDN 产生。

日志服务器验证源网站或 CDN 提交的操作的消息后, 会为源网站或 CDN 产生相应的接受声明, 声明在某一时刻前日志一定会在状态树中执行该操作。接受声明由日志服务器签名, 包含消息的内容、收到消息的时间、预计执行操作的时间。日志服务器接受的所有操作, 均会按照接受时间的先后顺序记录在操作记录中, 并在周期性更新时将操作结果体现在状态树中。

表 1 日志服务器支持的源网站和 CDN 的操作

Table 1 Supported operations of original websites and CDNs by public log

操作	消息内容	备注
源网站登记	域名, 消息类型, 代理摘要, 证书, 签名	源网站首次加入系统, 并提交代理摘要
源网站更新	域名, 消息类型, 新代理摘要, 证书, 签名	源网站更新代理摘要
CDN 登记	域名, 消息类型, 变色龙公钥, 证书, 签名	CDN 首次加入系统, 并提交变色龙公钥
CDN 更新	域名, 消息类型, 新变色龙公钥, 证书, 签名	CDN 更新变色龙公钥
源网站/CDN 注销	域名, 消息类型, 证书, 签名	源网站/CDN 离开系统

在详细介绍系统各方的交互之前, 我们先给出系统各主要流程的大致描述。源网站和 CDN 进入系统时, 首先在日志服务器中登记并记录自己的代理摘要和变色龙公钥。源网站可以在登记前预先生成代理摘要, 也可以在需要时生成并在日志中更新。源网站和 CDN 离开系统时, 需要在日志中注销自己的信息。

虽然本方案允许 CDN 更新其变色龙公钥, CDN 仍应严格保护它的变色龙哈希函数陷门, 尽量避免更新变色龙公钥的操作。因为 CDN 一旦变更变色龙公钥, 它代理的所有源网站均需更新代理关系, 造成较大的开销。事实上, CDN 的变色龙哈希函数陷门仅在它更新自己的 SSL/TLS 密钥时, 用来产生对旧公钥的碰撞, 在不改变源网站代理摘要的前提下独立地产生新的直接代理证据。因此, CDN 的变色龙哈希函数陷门应该离线保存, 避免泄露。

源网站生成代理摘要时, 会为相应的 CDN 生成直接/多步代理证据。每次日志服务器更新后, CDN 向日志服务器请求签名的树根, 以及 CDN 和客户源网站在新的状态树中的存在性证据。

浏览器访问连接被定向到 CDN 的代理服务器时, 代理服务器在 SSL/TLS 握手过程中发送浏览器所需的各类证据。浏览器验证通过后使用代理服务器的证书建立 SSL/TLS 连接。

源网站和 CDN 随时可以向日志服务器请求自己在状态树中的存在性证据, 检查状态树中记录的信息是否正确。同时, 审查机构检查唯一性和一致性证据, 并检查日志服务器是否正确运行。DET 中系统各方的通信过程如图 5 所示。我们在后续的小节中详

细介绍这些过程。

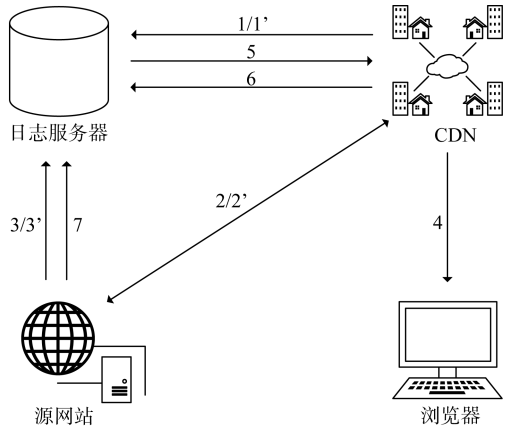

图 5　DET 系统中各方的通信过程

Figure 5　Basic communication flow in DET

4.2　源网站和 CDN 登记、注销、更新

当源网站或 CDN 在日志服务器中登记/注销后, 日志服务器将相应的节点加入/移除状态树。对于 CDN, **(步骤 1)**CDN 向日志服务器中发起登记操作, 以注册 CDN 的域名和变色龙公钥。CDN 在登记前应预先生成变色龙哈希函数陷门。对于源网站, **(步骤 2, 可选)**源网站提交登记请求前, 可以选择提前生成代理拓扑结构和代理摘要。源网站预先联系所有代理拓扑结构中的 CDN, 获取它们提供的 SSL/TLS 公钥、变色龙公钥和随机数 *r*, 并计算出代理摘要。源网站相应地计算直接/多步代理证据, 并提供给 CDN。源网站需要确认涉及的 CDN 均在状态树中出现, 且 CDN 在状态树中记录的变色龙公钥与提供的一致。源网站可以通过要求 CDN 提供 CDN 节点在状态树中的存在性证据完成这一过程。**(步骤 3)**之后, 源网站在日志服务器中登记并注册自己的域名和代理摘要。如果源网站在登记时没有授权任何代理, 可将代理摘要设为 NULL。**(步骤 6, 7)**一旦源网站和 CDN 请求注销, 日志服务器在更新时从状态树中删除对应的节点。

当源网站/CDN 的信息发生变化时, 如源网站产生新的代理摘要或 CDN 产生新的变色龙公钥, 它们需要在日志中更新。对于 CDN, **(步骤 1')**CDN 向日志服务器提交更新变色龙公钥的消息。CDN 的变色龙公钥更新后, 它代理的所有源网站都需要更新代理摘要。对于源网站, **(步骤 2')**源网站更新代理关系前, 首先计算新的代理摘要。此时源网站的行为同步骤 2。**(步骤 3')**源网站向日志服务器提交代理关系更新操作。日志服务器更新后, 方案依赖方开始使用新的代理关系摘要、变色龙公钥和代理证据。

4.3　浏览器验证

当浏览器通过 HTTPS 访问源网站时, 其连接可能被定向到源网站的服务器, 或 CDN 的代理服务器。在 DET 中, 源网站和 CDN 分别管理各自的 SSL/TLS 公私钥。因此, 浏览器收到的证书可能是源网站的证书(源网站服务器提供), 也可能是 CDN 的证书(CDN 代理服务器提供)。如果是前者, 浏览器正常地建立 HTTPS 连接; 如果是后者, 浏览器需要额外地验证代理服务器提供的关于代理关系的证据。

(步骤 4)当浏览器连接到 CDN 的代理服务器时, CDN 应在 SSL/TLS 协商过程中额外地提供关于代理关系的证据(使用 SSL/TLS 协议中握手消息的扩展), 以证明该 CDN 被源网站授权代理内容。完整的供浏览器验证的代理证据由以下四部分组成:

1) 最新的签名的树根;
2) 状态树中源网站的节点和存在性证据;
3) 状态树中 CDN 的节点和存在性证据;
4) 直接/多步代理证据。

浏览器首先验证签名的树根是否正确且未过期。浏览器进一步结合签名的树根中状态树的哈希, 验证源网站和 CDN 节点的存在性证据。浏览器使用源网站节点中的代理摘要, 和 CDN 节点中的变色龙公钥验证直接代理证据。在直接代理证据中, CDN 的域名、公钥和随机数 *r* 由 CDN 提供。CDN 提供的公钥应与浏览器收到的 CDN 证书的公钥一致。

浏览器建立连接时, 可以在 SSL/TLS 协议握手消息的扩展中声明请求完整的多步代理证据。浏览器验证多步代理证据的方式和验证直接代理证据类似。浏览器需要验证多步代理证据中其他 CDN 节点在状态树中的存在性证据, 得到它们的变色龙公钥, 并根据它们的域名、SSL/TLS 公钥和随机数(在代理证据中提供)额外地计算多步代理证据中这些 CDN 对应节点的哈希。

若上述验证通过, 则代理关系成立, 浏览器接受 SSL/TLS 连接, 并且明确得知连接的对端是已被授权的 CDN 代理服务器。若验证不通过, 浏览器终止 SSL/TLS 连接。

4.4　日志服务器更新

每个更新周期结束后, 日志服务器集中处理本周期内接受的所有操作, 并更新操作记录和状态树。更新完成后, 日志服务器发布新的签名的树根。

(步骤 5)此时, CDN 向日志服务器请求自己和客户源网站的节点在新的状态树中的存在性证据, 以便在 HTTPS 连接建立时向浏览器提供。

4.5 对日志服务器的监督和审查

源网站和 CDN 定期地向日志服务器请求关于自己的节点的内容和存在性证据, 并验证证据。一旦日志服务器在日志中恶意篡改源网站的代理摘要, 或 CDN 的变色龙公钥, 相应的源网站和 CDN 能够很快发现, 并采取额外的措施恢复。我们的实验表明, 日志中存在性证据的表示和验证是高效的。监督的过程不会为源网站或 CDN 带来过大的开销。

日志服务器、源网站、CDN、浏览器都可能向审查机构提交从各种渠道获得的签名的树根, 审查机构之间也会分享获得的签名的树根。审查机构比较这些签名的树根是否一致。如果审查机构发现在同一时刻有两个不同的签名的树根, 则说明日志服务器违反了了唯一性的要求。审查机构应公布不一致的签名的树根, 并对网络中各依赖方发出警告。

审查机构可以向日志服务器请求任意两个更新周期之间一致性证据。审查机构还可以下载特定时间段内公开日志的操作记录和对应的状态树, 检查 1)日志服务器进行的任何操作是否符合规则; 2)任何一个更新周期内, 日志服务器完成的操作和状态树的最终状态是否匹配。

5 安全分析

首先, 如果 DET 中各实体均正确运行, 在完整的代理证据验证通过后, 浏览器可以确认连接的服务器得到了源网站的授权。完整的代理证据中, 直接代理证据证明了 CDN 在源网站的代理拓扑结构中; 源网站的存在性证据, 表明了代理摘要是真实存在于日志服务器中的, 是经过源网站监督确认的; CDN 的存在性证据, 表明了计算所用的 CDN 变色龙公钥是真实存在于日志服务器中的, 是经过 CDN 监督确认的。只要源网站和 CDN 周期性地监督日志服务器, 确认日志服务器中的代理摘要正确, 未被授权的 CDN 就无法单方面产生有效的直接代理证据。因为它无法从一个有效的直接代理证据中, 找到某个 CDN 节点的哈希的碰撞。

未被授权的 CDN 可以和日志服务器合谋, 为一个虚假的代理摘要产生存在性证明, 并根据虚假的代理摘要产生虚假的直接代理证据。验证虚假的存在性证明时使用的签名的树根, 一定和真实的树根不同。如果收到虚假树根的浏览器在本更新周期内曾经收到真实的树根, 浏览器就会发现日志服务器违背了唯一性。此外, 浏览器可以定期将收到的签名的树根提交给任意审查机构。审查机构一旦收到虚假的树根, 就能发现同一更新周期内出现了不同的签名的树根, 从而发现日志服务器违反唯一性。

尤其地, 一旦虚假的树根出现, 出于一致性的要求, 日志服务器需要保证之后的任何时候均可给出当前树根到以前虚假的树根的一致性证据。由于哈希函数是单向的, 所以任何时刻与虚假树根满足一致性的树根与真实的树根会一直不同。此时一致性和唯一性无法同时满足。因此, 日志服务器产生虚假树根会永久地留下供审查的证据, 不会随时间消失。

合谋的日志服务器亦可能破坏预先设定的规则, 产生唯一的但是错误的日志。例如, 日志服务器可能在状态树中对某个特定源网站生成两个节点, 分别记录真实的和虚假的代理摘要。这样日志服务器可以向源网站提供真实代理摘要的存在性证明, 而向浏览器提供虚假代理摘要的存在性证明。由于审查机构会针对日志的每次更新下载日志的操作记录和状态树并进行检查, 因此可以发现日志服务器的此类作恶行为。

恶意的日志服务器可能拒绝服务, 拒绝特定源网站、CDN 或者审查机构的监视和审查。此时, 源网站或 CDN 可以委托任意其他审查机构代为监视日志。如果其他审查机构在一个更新周期中亦无法取得日志服务器的服务, 则认为日志服务器长时间违背可用性。

综合以上, 所有违反本方案要求的恶意攻击行为, 都会被其他实体或机构发现, 进而可以进一步地防止这些恶意攻击行为可能对浏览器造成的攻击。

6 实现和性能测试

6.1 系统实现

我们实现了 DET 的原型系统, 包括: (1)日志服务器, 支持源网站和 CDN 的各类操作, 产生基于密码学的证据; (2)支持 DET 证书验证方式的浏览器; (3)源网站管理工具, 支持向日志服务器请求登记、更新、注销操作, 产生代理摘要和直接代理证明, 监督日志服务器; (4)CDN 代理服务器(可以向浏览器发送完整的代理证据), 和管理工具(包括与日志服务器的所有交互功能); (5)审查工具, 请求并验证签名的树根、一致性证明, 和正确性检查。DET 原型系统中各个角色功能均由 C++实现。

支持 DET 方案的原型浏览器是在 Firefox Nightly(版本 54.0a1)的基础上实现的, CDN 代理服务器在 Nginx(版本 1.10.3)基础上实现的, 浏览器和服务器连接采用 TLS1.2 协议。具体地, 我们将 Firefox

中原有的证书验证方法 VerifySSLServerCert()修改为本方案中的浏览器验证过程。Firefox 浏览器支持在 ClientHello 消息中设定 Server Name Indication 扩展[9]。我们修改了协议消息格式, 为 ClientHello/SeverHello 消息定义了特定的扩展, 允许浏览器在访问 HTTPS 网站时使用 ClientHello 的消息扩展请求代理证据, CDN 代理服务器则将代理证据作为 SeverHello 的消息扩展和服务器证书一同发送给浏览器。源网站和 CDN 的管理工具支持登记、代理摘要/变色龙公钥的更新、注销、存在性证明的请求和验证。对于源网站管理工具, 还包括给定 CDN 域名、SSL/TLS 公钥和变色龙公钥时, 建立代理拓扑结构, 并为每个 CDN 产生代理证据的功能。在审查工具中, 我们实现了对日志服务器各类型证据的验证。

除去浏览器连接 CDN 代理服务器的过程, 各个实体间的消息通过 TCP Socket 的多线程队列进行传输。日志服务器持续地监听特定端口, 根据收到的请求类型, 向源网站/CDN/审查机构等相应地提供存在性证据、一致性证据、签名的树根、日志的操作记录和状态树等。其中日志服务器提供的操作记录编码为操作的序列, 状态树则做序列化处理。审查工具持续地监听特定端口, 接收或提供签名的树根。源网站/CDN 向日志服务器提交的操作的消息, 依照表 1 实现。

DET 原型系统实现了基于离散对数的变色龙哈希算法[8], 并使用 OpenSSL-1.0.1g 提供的 X.509 数字证书功能和密码算法, 包括哈希算法 SHA256 和签名算法 RSA-2048。

6.2 性能测试与分析

6.2.1 环境与数据设置

针对目前CDN提供HTTPS服务的实际规模, 我们考虑日志服务器的各类操作性能、浏览器验证代理证据的开销, 以及对日志监视和审查的开销。仿真实验结果表明, 在支持现有用户规模的情形下, 日志服务器的操作、浏览器的额外验证、对日志监视和审查均具有较小的开销。如果实际部署 DET 方案, 除了考察上述的各类性能外, 还需要考虑其他扩展性问题。例如, 日志服务器在高并发连接时的可用性、数据的冗余性备份和备份间的同步等。目前针对这类问题已经有若干技术方案[10-12]。

在仿真实验中, 我们设置 DET 日志中源网站的数量为 2 000 000, CDN 的数量为 10 000。文献[5]对全网证书进行了调查和测量, 指出这些证书的域名中, 与其他组织共用私钥的约为 196 万个。这可以作为使用 CDN 服务的源网站数量的大致上界。所以我们的设定与实际情况相符, 而且由下文的数据分析可以看出, DET 具备良好的可扩展性, 即性能开销未随着源网站数量增加而明显增长。我们假设平均每天有 1%的源网站登记、注销或更新。我们设日志服务器的更新周期为 2 小时。由此, 平均每次日志更新涉及 2 000 000×1%/12=1667 次操作。

在仿真实验中, 我们随机生成了 10 000 个 CDN 和 2 000 000 个源网站, 对每个源网站都随机生成了 2 层 CDN 级联的代理关系(如图 2), 并将生成的源网站和 CDN 加入日志。我们随机挑选了 1000 个源网站和 CDN, 并生成相应的登记/更新/注销操作的消息, 测试日志服务器执行这些操作的时间。测试操作记录更新和产生一致性证据的系统性能时, 我们随机挑选了 1667 个源网站并产生了总计 1667 个随机类型操作。测试浏览器性能时, 我们使浏览器连接 CDN 代理服务器, 并测试验证代理证据的时间。后续小节中每项性能评估均进行了 1000 次实验, 并统计平均时间开销。其中日志服务器的各类操作, 未包含源网站/CDN/审查机构建立连接并提交请求的时间。浏览器验证各类证据的时间, 也不包括建立连接和传输相关证据的时间。

日志服务器部署在台式机中(Intel i7-4770s/3.10GHz 处理器, 8GB 内存, ST-1000 DM003 硬盘, Win7 专业版操作系统)。浏览器、审查工具和源网站/CDN 管理工具均部署在另一台相同配置的台式机中。CDN 代理服务器部署在同样硬件配置的台式机中, 操作系统是 Ubuntu 16.04 LTS (32-bit)。各主机通过百兆局域网连接。

6.2.2 日志服务器性能

表 2 给出了日志服务器进行各类型操作的时间开销。可以看出, 源网站和 CDN 在日志服务器的平均操作耗时不超过 0.25 毫秒, 即每秒钟状态树可以完成 4000 次以上的操作。在状态树的节点插入、删除时, 我们考虑了二叉树的平衡。在状态树中的所有操作, 其时间复杂度为 $O(\log N)$, N 为源网站和 CDN 的数量。

每次日志更新需要的时间约为 393 毫秒, 包括在状态树中执行所有等待的操作、更新操作记录和生成新的签名的树根。可以看出, 日志服务器更新所需的时间很短, 对服务可用性几乎没有影响。更新操作记录的时间复杂度为 $O(M)$, 其中 M 是更新周期内的操作数。

日志服务器产生存在性证据平均需要 0.11 毫秒。每次更新后, 日志服务器产生全部源网站和 CDN 的新的存在性证据, 总耗时约为 220s。这是日志服务器

表 2 日志服务器各操作的时间开销(毫秒)

Table 2 The processing time of each type of operations of public log (in milliseconds)

操作	平均值	中位数	最大值	最小值
源网站/CDN 登记	0.23	0.23	0.41	0.18
源网站/CDN 更新	0.05	0.05	0.09	<0.01
源网站/CDN 注销	0.18	0.18	0.23	0.13
生成存在性证据	0.11	0.11	0.13	0.09
生成不存在性证据	0.23	0.22	0.29	0.20
操作记录更新	2.45	2.43	2.57	2.41
生成签名的树根	5.33	5.32	5.86	5.15
生成一致性证据	<0.02	<0.02	<0.02	<0.02

最大的时间开销。鉴于我们的实验是在普通个人台式计算机上完成的, 通过提升日志服务器性能, 所需时间会大幅降低。日志服务器产生一次存在性/不存在性证据的时间和空间复杂度是 $O(logN)$。

日志服务器生成一致性证据的时间小于 0.02 毫秒。一致性证据的时间和空间复杂度为 $O(KlogM)$, K 是一致性证据跨越的更新周期数目, M 是每个更新周期的操作数。

表 3 记录了由日志服务器产生的各类基于密码学的证据的大小, 最大约为 2.92KB, 不会对各实体的存储和通信造成影响。

表 3 日志服务器产生的各类证据大小(KB)

Table 3 The size of each type of proofs generated by public log (in kilobytes)

证据类型	平均大小
存在性/不存在性证据	1.59/2.92
签名的树根	0.33
一致性证据(相邻更新周期)	0.59

6.2.3 浏览器性能

测试浏览器通信开销时, 我们假设源网站的代理拓扑结构是三层的, 每层有两个 CDN 节点(如图 2)。浏览器访问的是第三层的 CDN。此时, 作为一次 HTTPS 连接的额外通信开销, 完整的代理证据的大小约为 3.86KB, 其中签名的树根约为 0.33KB, 存在性证据约为 3.18KB, 直接代理证据约为 0.35KB。注意到, 包含三级证书的证书链的一般大小为 4.02KB。

浏览器验证完整的代理证据所用的时间平均为 1.38 毫秒, 验证过程的各部分耗时也体现在表 4 中。浏览器验证证书所需的平均时间为 9.86 毫秒, 额外验证仅增加了 14%的时间。用户很难察觉其中的差别。因此浏览器端的开销不会是方案部署的显著障碍。验证多步代理证据的时间开销和具体的代理拓扑结构相关。对于计算能力较弱的移动终端, 可以通过简化的代理证据进一步减少时间开销(见 7.2 节)。

表 4 浏览器验证完整的代理证据时间开销(毫秒)

Table 4 The detailed processing time of validating delegation proof by the prototype browser (in milliseconds)

操作	平均值	中位数	最大值	最小值
验证签名的树根	0.15	0.15	0.16	0.14
验证存在性证据	0.11	0.11	0.13	0.09
验证直接代理证据	1.01	0.98	1.22	0.87
总计	1.38	1.36	1.58	1.23

6.2.4 监督与审查

源网站与 CDN 监督日志服务器, 验证关于自身节点的存在性证据, 其时间和通信开销和 6.2.3 节中验证存在性证据相同。

审查机构验证签名的树根的开销和浏览器部分相同。审查机构对于相邻周期间的一致性证据验证小于 1 毫秒。对于审查机构, 最大的开销在于检查日志的正确性, 这要求审查机构下载某个时间段内所有日志操作和整个状态树(复杂度为 $O(N)+O(M)$, N 为源网站和 CDN 的数量, M 为操作的数量)。这个过程是在浏览器的 HTTPS 连接之外进行的, 不影响 CDN 的 HTTPS 服务的用户体验。

综上, 我们的实验结果表明, DET 的日志服务器具有较高的效率。同时, DET 不会对浏览器引入过大的时间和通信开销。通过对时间和空间复杂度的分析, 我们发现 DET 具备良好的扩展性, 即随着源网站和 CDN 数量的增长, DET 主要操作的时间和空间开销未明显增长。

7 扩展与讨论

7.1 简化的代理证据

验证完整的代理证据时, 浏览器需要进行多次签名验证和哈希值计算。签名验证和哈希计算的数目由代理拓扑结构决定。对于资源特别受限或安全性要求较为宽松的浏览器(如移动终端中浏览器, 或访问不敏感网站时), 浏览器可以在 ClientHello 消息中要求 CDN 提供简化的代理证据, 以降低验证开销。简化的代理证据是由日志服务器签名的消息, 证明 CDN 被源网站授权:

简化的代理证据:=

<源网站域名, CDN 域名, CDN 的 SSL/TLS 公钥, 有效期, 日志的签名>

浏览器验证公开日志服务器的签名, 并确认其中 CDN 域名和公钥与收到的证书匹配, 且源网站域名

和浏览器的访问目标一致。如果验证通过, 浏览器接受 HTTPS 连接。

CDN 向日志服务器请求简化的代理证据时, 需要向日志服务器提供源网站的域名和相应的直接代理证据。日志服务器验证直接代理证据通过后签发简化的代理证据。

浏览器可将收到的简化的代理证据提交到审查机构。审查机构进一步向日志服务器申请完整的代理证据并验证, 将验证结果异步地返回给浏览器。如果审查机构验证不通过, 也会认为日志服务器运行不正确。此过程中, 浏览器向审查机构暴露了自身的访问行为; 同时, 审查机构对完整的代理证据的验证结果一般晚于 HTTPS 连接的建立。

7.2 防范转发环路攻击

转发环路(forwarding-loop attack)攻击[13]是由恶意源网站发起的针对 CDN 的攻击。在转发环路攻击中, 恶意源网站要求多个 CDN 互相请求关于源网站的内容, 从而构成环路。特别地, 由于 CDN 拥有大量的代理服务器, 对内容的请求可能由少量的代理服务器传播到更多的代理服务器, 产生急剧扩大的效应, 对 CDN 的可用性产生严重威胁。

转发环路攻击产生的根本原因在于, 在多个 CDN 同时存在时, CDN 获取源网站内容的路径缺乏无环约束。文献[13]认为的最可行的解决方案是, CDN 在 HTTP 头部特定字段记录内容的获取来源。下游CDN向上游CDN请求内容时, 需要检查内容的每一步获取路径, 避免环路的发生。然而, 该方案需要所有 CDN 合作, 才能防止转发环路出现。环路中任何一个恶意的 CDN 和源网站合谋, 都可能使得该方案失效。

DET 可以防止转发环路攻击。当下游 CDN 向上游 CDN 请求源网站内容前, 需要获得从上游 CDN 到自己的多步代理证据。下游 CDN 验证上游 CDN 和自己在代理拓扑结构中有直接的上下游关系。若验证不通过, 则下游CDN拒绝向上游CDN请求源网站内容。由于代理拓扑结构是无环的, 因此任何人(包括源网站)无法生成同时成立的、可构成环路的多步代理证据。注意, 本方案不需要所有 CDN 合作即可抵御转发环路攻击。

7.3 与证书透明化合作部署

证书透明化方案[7]建立了公开可审查的日志结构, 记录所有有效的证书。证书透明化的日志是仅可添加的, 一旦证书被添加到日志中就永远不会被删除。通过扩展证书透明化日志, 我们可以将对 DET 日志的操作保存在证书透明化的日志中。这时, DET 的日志服务器仅需维护状态树, 而将所有进行过的操作添加到证书透明化的日志中, 由证书透明化日志产生签名的树根。同时, 可以将证书透明化方案中的网站公钥和证书策略的信息加入到 DET 日志服务器的状态树中, 使得对网站公钥和证书策略的查询更为高效, 降低网站监督证书透明化方案日志的开销。另外, 证书透明化方案也可以保护 PKI 系统, 缓解虚假证书可能带来的危害。在 DET 中, 日志服务器和浏览器依赖 PKI 系统验证源网站和 CDN 的公钥。PKI 系统安全性的提升, 可以相应地提升我们方案的安全性。

8 相关文献

8.1 CDN 代理关系

2014 年, 文献[4]首次提出了 CDN 和源网站共用私钥的问题, 深入分析了 CDN 服务器为浏览器提供 HTTPS 服务的原理。2016 年, 文献[5]进一步广泛调查了数字证书主体与其他组织共用证书私钥的现状。文献认为, 至少 76.5%的证书中主体与其他组织共用私钥。

针对 CDN 和源网站共用私钥的问题, 已经有一些解决方案被提出。文献[4]探讨了名字限制证书(Name Constraint Certificate)方案, 允许源网站拥有名字限制证书。源网站可以使用名字限制证书进一步为 CDN 签发有效的数字证书, 所签发证书的主体限制为源网站域名。然而, 此方案的实用性较差, 由于: 1)CA 不太可能为源网站提供名字限制证书; 2)源网站需要实现 CA 的部分功能, 开销过大; 3)现有浏览器对名字限制证书的支持程度较低。

文献[4]提出了基于 DANE[14]的解决方案, 源网站可以在 DANE 的资源记录中声明目前在用的 CDN 和 CDN 的公钥。然而, 有研究[15]表明目前在所有 HTTPS 网站中, DANE 的部署率不足 0.05%。另外使用 DANE 会造成浏览器多次请求 DNSSEC 资源记录, 在一次 HTTPS 连接中可能导致长达数秒的时间开销[16]。

在 Keyless[17]方案中, CDN 不拥有源网站的私钥, 无法独立完成 SSL/TLS 会话密钥协商。SSL/TLS 握手过程中, CDN 将浏览器发送的部分消息转发至源网站, 在源网站的协助下产生会话密钥。这要求源网站参与到每次 HTTPS 建立过程中, 面临可伸缩性的问题, 违背了 CDN 带来的性能收益。另一方面, HTTPS 建立过程中 CDN 需要连接源网站, 造成额外延迟。

代理凭据(Delegated Credential)[18]方案中, 源网

站签发短期的代理凭据(包含 CDN 的名字和 SSL/TLS 公钥), 证明源网站到 CDN 的内容代理关系。代理凭据随 SSL/TLS 握手消息推送, 由浏览器验证。此方案中源网站无法撤销已经签发的凭据, 只能等待凭据失效。同时, 每当 CDN 更换 SSL/TLS 公钥, 或者凭据过期时, 源网站需要再次签发凭据, 造成较大的负担。另外, 在 CDN 级联的情形下, 凭据的格式无法表示源网站到被访问 CDN 的完整的内容代理路径。

OOB(Out-of-Band for CDNI)[19,20]方案中, 浏览器首先访问源网站的服务器, 获得源网站的资源映射文件。浏览器根据获得的资源映射文件, 访问指定的 CDN 获取源网站的内容。OOB 要求源网站为每个 HTTPS 连接提供映射文件, 违背了 CDN 带来的性能收益。同时, 一次访问中浏览器需要同源网站和 CDN 建立多个连接, 降低了用户体验。在 CDN 级联的情形下, 浏览器从多个 CDN 依次得到资源映射文件, 造成的延迟会更加严重。

Lurk(Limited-use-of-remote-keys)[21]和 STAR(Short-term, automatically renewed certificate)[22,23]方案中, 由源网站向 CA 申请, CA 周期性地签发短期证书(short-lived certificate), 并向 CDN 提供这些短期证书和私钥。一旦代理关系结束, CA 机构不再签发此类短期证书。这些方案虽然可以一定程度缓解代理关系的撤销问题, 但是由于短期证书依然还对应源网站的域名, 所以代理关系对浏览器依旧不透明。

综上, 已有的方案均无法同时实现浏览器可知、独立的密钥管理、主体可控的代理关系和性能高效可实用的要求。

8.2 公开日志相关应用

目前已有的基于公开日志的方案主要用于防范由虚假证书带来的各类攻击。虚假证书是一类由 CA 机构错误签发的证书, 将证书主体名字和不属于证书主体的公钥绑定起来。虚假证书可以被用来发动身份冒用攻击。

2013 年, 谷歌公司提出证书透明化方案[7], 旨在帮助发现虚假证书。证书透明化方案中, 浏览器验证证书时, 要求证书必须出现在公开的日志服务器中。网站定期查看公开日志服务器, 检查是否有关于自己的虚假证书。

与证书透明化方案不同, DET 主要实现对源网站内容代理关系的管理和验证。因此, 我们的方案中将代理关系, 而不是证书, 发布在日志服务器中。

证书透明化中, 日志是一个记录所有证书的 Merkle 哈希树(对应于 DET 中操作记录的部分)。因此, 检查日志中某个网站的证书需要遍历日志中所有证书。为了提高查找效率, CIRT[24]提出了 Merkle 哈希树和搜索树结合的结构。CIRT 中, 查询一个网站的证书, 只需进行 *O(logN)*级别的操作(*N* 为网站数量)。我们的方案部分沿用了 CIRT 的日志结构。

AKI[25], ARPKI[26], Policert[27]将证书和证书策略加入公开的日志中。证书策略是由证书主体声明的验证证书的额外要求。证书策略可以帮助平衡 CA 的绝对权威, 实现对 CA 信任范围的限定。将证书加入日志服务器前, 或浏览器验证证书时, 要检查该证书是否满足主体的证书策略。这些方案, 在实现证书透明化的同时, 进一步限制了 CA 签发证书的范围, 增加了敌手产生虚假证书的难度。

PKISN[28]将证书被签发的时间和被撤销的时间也记录在日志服务器中, 实现了时间维度上细粒度的证书撤销管理。Policert、PKISN 均采用了与 CIRT 类似的结构。

CONIKS[29]是一个供身份服务商记录用户 ID 和公钥的日志系统。CONIKS 实现了对用户公钥信息的隐私保护: 知道用户 ID, 可以在 CONIKS 中查询 ID 对应的公钥; 不知道用户 ID, 则无法从 CONIKS 中得到用户的信息。CONIKS 使用前缀树作为日志的结构, 树节点由可验证随机函数产生(Verifiable random functions)。我们的方案可以结合 CONIKS 的方法生成状态树, 实现日志中源网站和 CDN 信息的隐私保护。Insynd[30]相应地提出了一种新的公开日志系统中保护用户隐私的密码学方案。

9 结论

本文提出了 DET, 一种基于公开可审查的日志方案, 解决目前 CDN 提供 HTTPS 服务时源网站私钥存在泄露风险、源网站无法独立且即时撤销代理关系、代理关系对浏览器不透明等问题。DET 使得 CDN 不必要求源网站分享私钥, 就可以为浏览器提供 HTTPS 服务; 浏览器可以明确获知并验证源网站与 CDN 之间的内容代理关系; 源网站独立地管理自己的内容代理关系。

在 DET 中, 日志的所有操作都保存在仅可添加的操作记录中。日志维护状态树, 提供高效的查询服务。日志提供基于密码学的证据, 使得审查机构能够检查其是否正确运行。

DET 仅需对 SSL/TLS 握手过程进行扩展, 加入代理证据的验证, 与现有的 TLS/PKI 体系兼容。DET 的性能评估表明 DET 的各项操作均具有较高的效率, 不会对浏览器建立 HTTPS 过程带来过大的时间(小

于 1.58 毫秒)和通信开销(约为 3.86KB)。

参考文献

[1] “Content Delivery Network (CDN) Market by Type (Standard/Non-video and Video), Core Solution (Web Performance Optimization, Media Delivery, and Cloud Security), Adjacent Services, Service Providers, and Region - Global Forecast to 2021”, http://www.marketsandmarkets.com/Market-Reports/content-delivery-networks-cdn-market-657.html, Nov, 2016.

[2] “Content Delivery Network (CDN) Market worth $12.16 Billion by 2019”, http://www.marketwatch.com/story/content-delivery-network-cdn-market-worth-1216-billion-by-2019-2014-03-27, Mar, 2014.

[3] “Global Content Delivery Network Market (2016–2022)”, https://www.giiresearch.com/report/kbv406073-global-content-delivery-network-market.html, Nov, 2016.

[4] J. Liang, J. Jiang, H. Duan, K. Li, T. Wan, and J. Wu, “When HTTPS Meets CDN: A Case of Authentication in Delegated Service,” in Proc. of *IEEE Symp. Security and Privacy (SP’05)*, pp. 67-82, 2014.

[5] F. Cangialosi, T. Chung, D. Choffnes, D. Levin, B. Maggs, and A. Mislove, et al, “Measurement and Analysis of Private Key Sharing in the HTTPS Ecosystem,” in Proc. of *ACM Sigsac Conference on Computer and Communications Security (CCS’16)*, pp. 628-640, 2016.

[6] B. Niven-Jenkins, F. Faucheur, and N. Bitar, “Content Distribution Network Interconnection (CDNI) Problem Statement,” RFC6707, IETF, 2016.

[7] B. Laurie, A. Langley, E. Kasper, “Certificate Transparency,” RFC6962, IETF, 2013.

[8] X. Chen, F. Zhang, and K. Kim, “Chameleon hashing without key exposure,” in Lecture Notes in Computer Science, pp. 87–98, vol. 3225, 2004.

[9] S. Blake-Wilson, M. Nystrom, D. Hopwood, J. Mikkelsen, and T. Wright, “Transport Layer Security (TLS) Extensions”, RFC3546, IETF, 2003.

[10] K.M. Chandy, and L. Lamport, “Distributed Snapshots: Determining Global States of a Distributed System,” in *ACM Transactions on Computer Systems*, pp. 63-75, vol. 3, 2016.

[11] K. Shvachko, H. Kuang, S. Radia, R. Chansler, “The Hadoop Distributed File System,” in Symposium on *MASS Storage Systems and Technologies*, pp. 1-10, 2010.

[12] F. Chang, J. Dean, S. Ghemawat, W.C. Hsieh, D.A. Wallach, and M. Burrows, “Bigtable: A Distributed Storage System for Structured Data,” in Symposium on Operating Systems Design and Implementation (OSDI’06), pp. 205-218, vol. 26, 2006.

[13] J. Chen, J. Jiang, X. Zheng, H. Duan, J. Liang, and K. Li, et al, “Forwarding-Loop Attacks in Content Delivery Networks,” in Proc. of *Network and Distributed System Security Symposium (NDSS’16)*, 2016.

[14] P. Hoffman, J. Schlyter, and K. AB, “The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA ”, RFC6698, IETF, 2012.

[15] P. Szalachowski, and A. Perrig, “On Deployment of DNS-based Security Enhancements,” in Proc.of *Financial Crypto and Data Security (FC’17)*, 2017.

[16] R. L.Barnes, “DANE: Taking TLS Authentication to the Next Level Using DNSSEC”, IETF Journal, Oct, 2011.

[17] N. Sullivan, “Keyless SSL: The Nitty Gritty Technical Details”, https://blog.cloudflare.com/keyless-ssl-the-nitty-gritty-technical-details/, Sep, 2014.

[18] R. Barnes, S. Iyengar, N. Sullivan, E. Rescorla, “Delegated Credentials for TLS,” Technical Report, IETF, 2017.

[19] J. Reschke, and S. Loreto, “‘Out-Of-Band’ Content Coding for HTTP,” Technical Report, IETF, June, 2017.

[20] F. Fieau, E. Stephan, S. Mishra, “HTTPS delegation in CDNI,” Technical Report, IETF, 2017.

[21] Y. Sheffer, “Delegating TLS Certificates to a CDN,” Technical Report, IETF, 2017.

[22] Y. Sheffer, D. Lopez, A. Perales, T. Fossati, “Use of Short-Term, Automatically-Renewed (STAR) Certificates to Delegate Authority over Web Sites,” Technical Report, IETF, 2017.

[23] F. Fieau, E. Stephan, S. Mishra, “CDNI interfaces update for HTTPS delegation,” Technical Report, IETF, 2017.

[24] M.D. Ryan, “Enhanced Certificate Transparency and End-to-End Encrypted Mail,” in Proc. of *Network and Distributed System Security Symposium (NDSS’14)*, 2014.

[25] T. Kim, L. Huang, A. Perrig, C. Jackson, and V. Gligor, “Accountable Key Infrastructure (AKI): A Proposal for a Public-key Validation Infrastructure,” in Proc. of *International Conference on World Wide Web (WWW’13)*, pp. 679 – 690, 2013.

[26] D. Basin, C. Cremers, H. Kim, A. Perrig, R. Sasse, and P. Szalachowski, “ARPKI: Attack Resilient Public-Key Infrastructure,” in Proc. of *ACM Conference on Computer and Communications Security (CCS’14)*, pp. 382–393, 2014.

[27] P. Szalachowski, S. Matsumoto, and A. Perrig, “PoliCert: Secure and Flexible TLS Certificate Management,” in Proc. of *ACM Conference on Computer and Communications Security (CCS’14)*, pp. 406–417, 2014.

[28] P. Szalachowski, L. Chuat, and A. Perrig, “PKI Safety Net (PKISN): Addressing the Too-Big-to-Be-Revoked Problem of the TLS Ecosystem,” in Proc. of *IEEE European Symposium on Security and Privacy (EuroS&P’16)*, pp.407-422, 2016.

[29] M. Melara, A. Blankstein, J. Bonneau, E. Felten, and M. Freedman, “CONIKS: Bringing Key Transparency to End Users,” in Proc. of *USENIX Conference on Security Symposium, (USS’15)*, pp. 383–398, 2015.

[30] R. Peeters, and T. Pulls, “Insynd: Improved Privacy-Preserving Transparency Logging,” in Proc. of *European Symposium on Research in Computer Security (Esorics’16)*, pp.121-139, 2016.

王泽 于 2013 年在莱斯大学电子信息工程专业获得工学硕士学位。现在中国科学院信息工程研究所/中国科学院大学网络空间安全学院信息安全专业攻读博士学位。研究领域为网络安全。研究兴趣包括: PKI、DNS、网络身份管理、区块链技术安全等。Email: wangze@iie.ac.cn

蔡权伟 于 2015 年在中国科学院信息工程研究所信息安全专业获得工学博士学位。现任中国科学院信息工程研究所助理研究员。研究领域为网络与系统安全。研究兴趣包括: 分布式容错系统、web 安全、网络身份管理等。Email: caiquanwei@iie.ac.cn

李文强 于 2016 年在华中科技大学软件工程专业获得工学学士学位。现在中国科学院信息工程研究所/中国科学院大学网络空间安全学院网络空间安全专业攻读硕士学位。研究领域包括: 网络与系统安全。研究兴趣包括: 虚拟化安全、网络身份管理等。Email: liwenqiang@iie.ac.cn