Deploying Cryptography in Internet-Scale Systems: A Case Study on DNSSEC

Hao Yang[1], Eric Osterweil[2], Dan Massey[3], Senior Member, IEEE,
Songwu Lu[2], and Lixia Zhang[2], Fellow, IEEE
Nokia[1], UCLA[2], Colorado State University(科罗拉多州立大学)[3]

IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING(TDSC) 2011 CCF-A

Seddon

2024-05-31

目录

文章简介

DNS 安全扩展（DNSSEC）是在互联网运行系统中部署加密保护的首次尝试。DNSSEC应用成熟的公钥加密技术来确保DNS系统中的数据完整性和来源真实性。尽管DNSSEC的加密设计合理且看似简单，但其开发却耗费了IETF十多年的时间和数次协议修订，即使在今天(2011)，其部署仍处于早期推广阶段。

本文首次系统地探讨了DNSSEC多年来在设计、部署和运行方面遇到的挑战。我们的研究揭示了加密设计与互联网运行系统之间的根本差距。要在全球互联网中部署，加密协议必须具备几个关键特性，包括可扩展性、灵活性、可增量部署性，以及在不完善的运行条件下发挥作用的能力。

本文希望从这项研究中获得的见解能为未来其他互联网规模系统的加密设计提供宝贵的意见。

文章简介

许多挑战来自几个基本因素：

- 运行中的DNS系统的大规模和分布式性质
- DNS数据缓存的存在
- 不同自治管理机构在DNS运行中固有的异质性

本文的主要贡献有三方面：

1. 对DNSSEC部署和运行中的挑战进行了记录和分类
2. 分析了DNSSEC社区为应对这些运行挑战所做的不懈努力，并针对一些开放性问题提出了我们自己的解决方案
3. 将DNSSEC的演变总结为一系列设计经验，希望对互联网上部署的其他加密系统的设计有所帮助

2 DNS和DNSSEC

DNS：

- RR（Resource Record）： <name, class, type>
- RRSet（Resource Record Set）：拥有相同name, class, type的RR的集合
- Zone
- 胶水记录（glue records）：域管理者向上级域管理者提供的一组主机名和IP的映射表
- 名称解析器和本地缓存

全局DNS是一个以树形结构组织的分布式数据库。在树的顶部，根域将权限委托给顶级域，例如.com、.net、.org和.edu。然后，.com域将权限委托给createibm.com，.edu域将权限委托给创建ucla.edu，等等。组成域数据库的信息存储库分为称为域（Zone），每个域都属于一个管理机构。

2 DNS和DNSSEC

DNSSEC：

- 保证数据完整性和来源真实性
- 新的RR类型DNSKEY RR 和RRSIG
- 每个签名都带有到期时间戳
- 为了验证密钥，DNSSEC利用现有的DNS委托层次结构来提供公钥基础设施(PKI)。在此PKI中，每个父区域都会签署其子区域的DNSKEY RR，而根区域的公钥则以安全的**带外机制**分发给所有解析器
- 例如，根公钥用于验证org公钥，org公钥又用于验证foo.org公钥...
- 使用NSFC记录间接证明某些解析记录不存在
- 2005年5月正式确定了DNSSEC的规范，并试点在了.org .se .br等域名中

3. 在分层PKI上的设计问题

PKI（公钥基础设施 Public-Key Infrastructure）

- DNSSEC 通过利用现有的 DNS 委托层次结构构建公钥基础设施
- 每个域都会签署其所有子域的公钥

而这个看似简单的PKI设计在实施和开始部署时却遇到了四个意想不到的问题：

- 可扩展性
- 跨边界数据处理
- 跨域协调
- 增量部署能力

在分层PKI上的设计问题
3.1 可扩展性

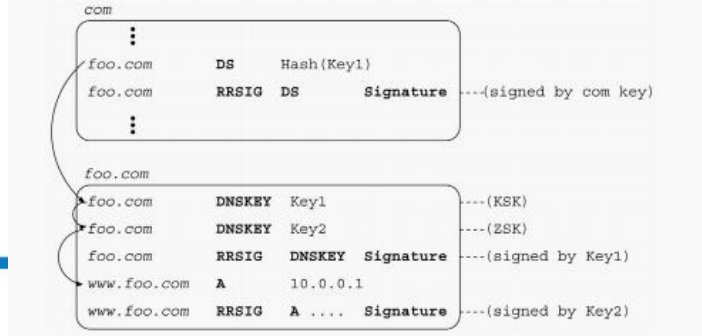

Fig. 3. One level of indirection is added in the trust chain through DS RR, KSK, and ZSK.

DNSSEC通过验证其父区域的相应签名来验证区域的公钥。

- 第一个 DNSSEC 规范将区域公钥及其父级签名存储在（子）区域，但这种简单的设计决策忽略了一个重要因素：
  - DNS域可能有大量子区域。如：.com 域拥有数千万个委派子域，每当区域更改其密钥时，必须联系每个子区域更新公钥签名
  - 当父区域更改公钥时，每个子域都必须联系父区获取更新的签名，父区域将负载大量更新请求
- 随后的 DNSSEC 规范做出了两项更改来解决上述问题扩展问题。
  - 1. 定义一个新的委托签名者 (Delegation Signer, DS) RR，用于在父区域存储子域KSK公钥的哈希值，并由父密钥签名（存储子域的公钥哈希值）
    - 当一个域更改其密钥时， 自行重新签署本地存储的所有子区域的 DS RR即可，而无需通知任何子区域
  - 2. 允许一个区域拥有多个公钥
    - 密钥签名密钥或 KSK（签署其他密钥）和区域签名密钥或 ZSK（签署区域数据）
    - KSK 公钥作为 DS RR 存储在父区域中。 KSK私钥用于对 DNSKEY RR 集 （包括 KSK 和 ZSK）进行签名。
    - ZSK 用于签署区域的数据记录，这些签名通过以下方式验证：
      - 首先验证区域的 KSK（图中Key1）
      - 然后用 KSK 验证包含 ZSK 的 DNSKEY RRset 的签名
      - 最后用 ZSK 验证数据的签名
  - 由于 KSK 与 ZSK 分离，区域可以在本地更改其任何 ZSK，只有在其 KSK 更改时才会联系其父区域更新 DS RR
    - 由于 KSK 只用于签署 DNSKEY RRset，因此它也可以得到更好的保护（例如，保持离线），并减少受到密码分析攻击的可能性
    - 例如，Kolkman 和 Gieben [13] 建议一个区域每月更换一次 ZSK，每年更换一次 KSK

从密码学的角度来看，原始 DNSSEC 规范和目前的修订版提供了相同级别的密码保护。

从系统角度看，后来的两处改动比最初的设计在可扩展性方面有了显著改进，使其具有部署的可行性。

在分层PKI上的设计问题
3.2 跨边界数据处理（未签名的DNS数据）

DNS提供了严格的数据所有权定义:

- 每个 RRset 属于一个且仅一个区域，每个区域都会签署其所有权威数据，包括来自其所有子区域的DS RR
- 每个区域的 NS RR 和glue A RR 都存储在本地区域和父区域中，以便 DNS 自上而下查询
- 由于 NS 和glue RR 集被定义为子区域的权威数据， 因此 DNSSEC 设计使它们在父区域中未签名
- 即使中间人可以伪造虚假委托 NS RR 以 将 DNS 查询误导到攻击者选择的主机，攻击者仍然无法伪造子区域中的任何 DNS 数据（没有私钥）

虽然确保了DNS层级结构中的密钥和记录的真实性和完整性，但存在问题:

攻击者获得了 foo.com 区域的**私钥**，但没有获得对该区域 DNS 服务器的控制：攻击者可以伪造 从.com 的引用，并将 foo.com 的查询重定向 到他选择的主机。由于攻击者知道 foo.com 的私钥， 因此伪造的答案将被 DNS 解析器接受。

如果 .com 区域对委托 NS RR进行签名，那么攻击者首先就无法伪造引用来重定向查询。此示例表明，让父区域签署其子NS RR 可以在子代私钥泄露的情况下带来额外的保护。这种签名可以通过当今的 DNS 实践轻松实 现，因为子区域需要将其名称服务器中的任何更改通知其父区域（主要通过手动过程）。

实际上不需要机械地遵守数据所有权规则，要做好对关键基础设施的保护。

在分层PKI上的设计问题
3.3 跨域协调

DNS通过协调父区域和子区域之间的 NS 和glue A 记录来维护名称委托链

- 子区域对这些记录所做的任何更改都必须及时传播给父区域，这种协调是由人类手动完成的，非常容易出错。
- 至少 15% 的区域遭受跨越子区域和父区域边界的配置错误[26]
- DNS 可以通过DNS服务器的冗余来容忍这种不完美的父子协调，并且只要父区域至少知道子区域使用的一台正确的权威服务器， DNS 名称解析就可以成功。

- DNSSEC 所需的跨域协调要求更高：
  - 每个父子对不仅需要协调 DNS 服务器更改，还需要协调定期密钥更改。
  - DNS ：只要父区域至少知道子区域的一个正确名称服务器，委派链接就可以工作，并且当子区域拥有更多 DNS 服务器时， 成功的机会就会增加。
  - DNSSEC：只要攻击者破坏了与 DS RR 匹配的一个公钥，DNSSEC 就会失败，因此拥有多个 DS RR 可能会降低而保护强度。
- [1] 中已经发现了大量的DNSSEC配置出错， [25][12]中表明出现了许多 DNS 监控工具和服务来检查跨域配置错误

在分层PKI上的设计问题
3.4 增量部署能力

- **只有Z的全部祖先部署了DNSSEC，才能形成从根到Z的认证链**
  - 鉴于 DNS 是由数百万个自治管理域维护的分布式系统，当各域独立打开DNSSEC时，会形成多个安全孤岛
  - 每个子树根的公钥称为信任锚，由于信任锚 KSK 无法被未部署 DNSSEC 的父区域验证，因此解析器需要通过其他方式来收集、验证和维护信任锚 KSK
  - DNSSEC 设计没有提供解析器从子树中获取 KSK 信任锚的安全机制
- 当前指南建议**手动配置**，虽然手动配置方法可以在小规模下工作，但其可行性随着部署基础规模的增加而降低。由于每当信任锚 KSK 发生变化时，每个缓存解析器都必须更新其配置文件，因此当部署基础增长时，**这种方法将面临支持 DNSSEC 的缓存解析器数量和信任锚数量方面的可扩展性问题。**
  - 一些提案 [15]、[17]克服了手动配置所有信任锚的困难。然而这些提案的具体细节和安全分析却缺失。
  - 最近的一项提案 DNSSEC 旁视验证 (DLV) 建议由 DNSSEC 系统外部的**受信任机构（例如 VeriSign 或 ISC）**签署区域的公钥。
- 总体而言，DNSSEC 缺乏增量部署的配置严重阻碍了其部署，并且迄今为止所有在原始设计中添 加各种补丁的尝试尚未被证明是有效的。本文作者提出： **分布式监控框架可以为 DNSSEC 增量部署提供有效的解决方案。**

4 DNS缓存的影响

存根解析器直接与缓存通信，间接与权威服务器通信

- Alice（存根解析器）想要验证来自Bob（名称服务器）的数据
  - Alice 无法直接与 Bob 通信。
  - Alice只能向代表**缓存**的第四个玩家Carol 发送消息。
  - Carol 可以使用 Carol 认为正确的缓存数据来回答问 题，也可以联系 Bob 以获得答案。
  - 在正常操 作期间，Bob 可以更新数据并刷新签名、更改公 钥、撤销公钥等等。但Bob所做的任何更改对于Alice来说都不会立即可见。
  - 在Bob替换数据或密钥之后 很长时间，Alice可能会继续从Carol 接收缓存数 据。

DNS缓存的影响
4.1 密钥轮转

DNSSEC建议密钥需要随着时间的推移而更改，密钥轮换的目标是逐步淘汰旧密钥并用新密钥替换

- 由于缓存的存在，从名称服务器中删除旧的DNSKEY RR（或DS RR） 并不会将其从系统中删除
- **这些缓存的条目仅在其TTL或签名过期后才会被删除，忽略缓存的影响可能会破坏身份验证链**
- 例如，存根解析器可以从缓存接收旧签名并查询密钥以验证签名。如果缓存解析器没有密钥并且权威服务器回复了新密钥，则存根解析器无法验证缓存数据，因此必须拒绝它

操作指南[13]在密钥更新过程中为旧密钥引入了**一个宽限期**，考虑区域更改其KSK的场景：

1. 将新KSK添加到其DNSKEY RR集中，**但保留旧KSK以保留身份验证链。**
2. 旧的DNSKEY RR 集超时，则缓存要么没有 DNSKEY RR 集，要么具有同时包含新旧 KSK 的 DNSKEY RR 集。
3. 此时，**父级的 DS RR 发生更改以匹配新的 KSK，身份验证链仍会保留。**因为具有旧 DS RR 的任何缓存都将与旧 KSK 匹配，并且具 有新 DS RR 的任何缓存都将与新 KSK 匹配。该区域再次等待，直到所有缓存的 DS RR 过期。此时，所有 缓存要么没有 DS RR，要么有新的 DS RR。
4. 从 DNSKEY RR 集中删除旧的 KSK，并且通过新的 KSK/DS RR 的新身份验证链开始生效。

RFC5011 [30] 使用多个密钥和多个 DS 记录。在此方法中，区 域可以发布活动 KSK 和备用 KSK。两个 KSK 都有存 储在父区域中的相应 DS 记录。在密钥轮转中，活动 KSK 退役，备用 KSK 变为活动，并添加一个新的 KSK 作为新的备用 KSK。除了更改区域密钥的复杂性 之外，运营商还需要考虑可能需要演进密钥正在使用 的加密算法，例如从 RSA 到 GOST。管理此类加密算 法轮转的过程尚未完全指定，但它们可能会遵循用于 密钥轮转的类似步骤。

DNS缓存的影响
4.2 密钥撤销

如果发生密钥泄露事件（"紧急密钥翻转"），则攻击者可以继续重放旧密钥，直到相关签名到期，这可能需要几天或甚至一个月。

RFC5011[30]中的多密钥翻转过程同时使用活动密钥和备用密钥。如果只有活动密钥被泄露，则可以撤销该密钥并立即切换到备用密钥。同样，如果只有备用密钥被泄露，则可以立即撤销备用密钥。**不幸的是，直到 2007 年末RFC5011发布，DNSSEC 规范才提供明确的密钥撤销机制，目前选择实施这一新规范的工作由各个区域运营商负责。**（且如管理com域的TLDs并不打算实施RFC5011）

在缺乏通用密钥撤销支持的情况下，无论区域操作员采取什么操作，攻击者始终可以重放受损密钥并使用它 成功伪造 DNS 记录，直到其签名生命周期到期。

RFC4641[13] 中指出：**"如果滥用受损密钥比缓存数据无法验证更糟糕，则区域运营商必须做出权衡。"**

DNS缓存的影响
4.3 缓存存根的验证策略

随着 DNSSEC 普及，一个自然的选择是缓存解析器来处理构建身份验证链和验证所有接收到的数据的复杂性。

- 如果存根解析器不信任缓存服务器，则此方法会带来安全问题
- 例如公共场所互联网提供的缓存解析器，用户不知道缓存解析器是否配置有期望的信任锚和安全策略。不同的配置可能会导致缓存解析器和存根解析器之间的验证冲突
- 为了减少存根解析器对缓存解析器的依赖，DNSSEC允许存根解析器通过查询中的检查禁用 (CD) 位强制执行其自己的本地策略。当缓存解析器收到带有 CD 位的查询时，它应该将答案转发给存根解析器而不执行验证（相当于只用作缓存）**解决了漏报， 但难以解决误报**

存根解析器不能刷新缓存解析器的缓存，缓存解析器会返回相同的答案，直到 TTL 过期。存根解析器获得不同答案的唯一方法是使用**另一个缓存解析器或直接解析查询本身**

DNS 严重依赖缓存来减少服务器负载并提高查询解析性能， 并隐含假设存根解析器信任它们默认的任何缓存解析器。

DNSSEC 暴露了存根和缓存解析器之间的信任关系问题。存根解析器现在必须做出明确的决定，是自行执行验证（这会破坏 DNS 缓存），还是进行配置。

DNS缓存的影响
4.4 缓存同步

➢DNSSEC要求TTL和签名哪个先到期就清除Cache

- TTL失效时间不同，而签名到期的时间通常是相同的
- 造成“缓存同步效应”，签名到期时，将产生大量查询请求
- 假设查询遵循破泊松过程（到达率为λ，且此时忽略签名到期），则引发缓存未命中的概率为P=1/(1+λ*TTL), 于是平均查询率为 $\frac{\lambda}{1+\lambda \cdot TTL}$
- 如果签名到期，则峰值负载与平均负载之间的比率： $$\gamma = \frac{\lambda}{\frac{\lambda}{1+\lambda \cdot TTL}} = 1 + \lambda \cdot TTL. \quad (1)$$
- 这种意外的同步行为不是DNSSEC 签名生命周期独有
  - 使用 ACK 或 NAK 的简单可靠多播 设计可以在成功传送后在源处触发 ACK（或 NAK）爆炸或丢包
- **协议设计必须避免在大规模分布式系统中触发同步操作**

DNS缓存的影响
4.4 缓存同步

➢**缓存同步的解决方案**

1. 通过使 RRset 的 TTL 在伴随签名之前过期：
   - RFC4641[13] 要求区域运营商在签名过期时间之前的一个 TTL 之前进行替换
   - 区域运营商可能会忽略此准则或意外忘记及时更新签名，或错误的时钟.....
2. **在缓存解析器上执行TTL调整**
   - 当缓存解析器收到 RRset 时，它首先检查 TTL 或签名是否过期
   - 设 T_r表示接收到 RRset 的时间，T_e 表示 签名到期时间
   - 如果 T_r + TTL> T_e （即签名先过期），则缓存解析器将 TTL 修剪为范围内的随机值[0, T_e - T_r]；

5 异构操作的影响

- DNSSEC 操作中的另一个重要问题是域应该将其**私钥保持在线还是离线**
  - DNSSEC域必须确保其私钥的保密性，并使用这些密钥对其所有权威 RRset 进行签名
  - 保持密钥在线（根据签名动态生成或 更改的数据的要求）与保持密钥离线以更好地保护的愿望之间存在根本冲突
  - 两个问题：存在否认和安全动态更新
- **离线私钥**更受欢迎，例如存储在称为签名者的非联网且物理安全的计算机中
  - 当由于私钥无法在线访问，因此其暴露的机会大大降低
  - RFC 4034[4]建议使用离线密钥的做法，但离线密钥使得对动态更新进行签名变得困难，每次需要对一条新数据进行签名时都不可能调用离线签名器
- **在线私钥**可以轻松签署动态更新，但面临更大的安全风险
  - 将私钥放在多个 DNS 服务器上可 能会带来很高的安全风险。在全球DNS中，每个区域都 部署冗余的权威服务器，也称为辅助服务器。为了最大 限度地提高服务可靠性，建议将冗余服务器放置在拓扑 不同的位置，即由区域所有者以外的管理员管理的位置。因此，将密钥保持在线会导致很大的错误空间
  - 单个私钥的泄露可能会产生多米诺骨牌效应
    - 在 DNSSEC 中，区域的私钥不仅用于对该区域中的 DNS 数据进行签名，还用于所有子区域的委派
    - 一个区域密 钥的泄露可能会导致连锁效应，从而导致多个 DNS 区 域受到泄露

异构操作的影响
5.1 处理动态更新

- 由于DHCP的广泛使用，服务器的IP可能是是动态的
  - A记录需要频繁更新
  - 引入DNSSEC后则需要更新相关记录并签名
  - 所有这些新记录必须立即签署
  - 如果区域密钥保持离线状态，则 DNS 运营商必须调用离线签名，这违背了自动更新机制的目的

当前的 DNSSEC 规范（RFC 3833）承认此冲突：**"区域签名密钥必须可用于创建签名的 RR 集以放置在更新的区域中。该密钥必须在线（或至少可用），这一事实存在潜在的安全风险。"**

请注意，这一冲突的解决不是一个理论上的问 题，而是一个操作问题，只有精心设计才能在实践中找到良好的解决方案。

异构操作的影响
5.2 验证是否存在

DNSSEC 需要为**请求不存在记录的查询时提供**经验证的答案，当 DNS 以"不存在的记录"答复 查询时，服务器可以使用密钥来构建和签署不存在证明，但需要保持区域的私钥在线。

如需私钥离线，则该区域必须先构建并签署不存 在的证明。最新 DNSSEC 规范 [3]中 的 NSEC 方案采用了这种方法：

- **产生"区域行走"问题，可能造成隐私泄漏**
- 存储成本较高，添加NSEC RRs大约会使区域文件大小增加一倍

- 解决方法：
  1. RFC4470提案假设私钥在线，当服务器接收到对不存在的名称 （QName）的查询时，它会生成并签署一个最小覆盖的 NSEC RR。
  2. NSEC3对所有现 有名称进行哈希处理，并将这些哈希值（与原始名称相 反）排序为链。它为哈希值链中的每一跳生成一个 NSEC RR，即每个 NSEC RR 中的所有者和下一个名称 是两个相邻的哈希值。
     - 提高了协议复杂度
     - 也会造成区域的文件变大

6 DNSSEC监控和部署状态

- 使用SecSpider[12]对DNSSEC的部署进行监控
  - 一个分布式轮询系统，在本文撰写时(2011)，已经于美洲、欧洲和亚洲部署
  - 此时估算约10523个启动DNSSEC的区域，约1000个生产环境
  - 根据有无MX和WWW记录判断是否为生产环境
  - 目前SecSpider观测到了约730个孤立的安全岛（95.5%的深度为1）

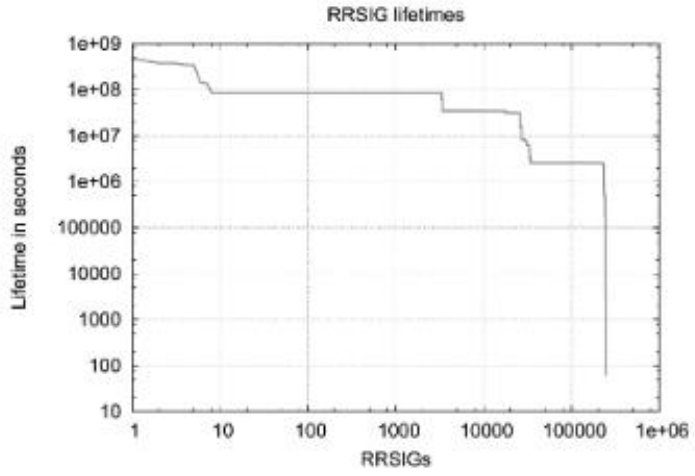

Fig. 4. This Figure is drawn in log-log scale and shows the distribution of the RRSIG lifetimes of RRsets in the production zones.

RRSIGs生命周期

较大的DNSSEC岛并不代表DNSSEC的使用率增加，因为它们可能只是由一个运营组管理的多个区域组成，而不是多个独立方。

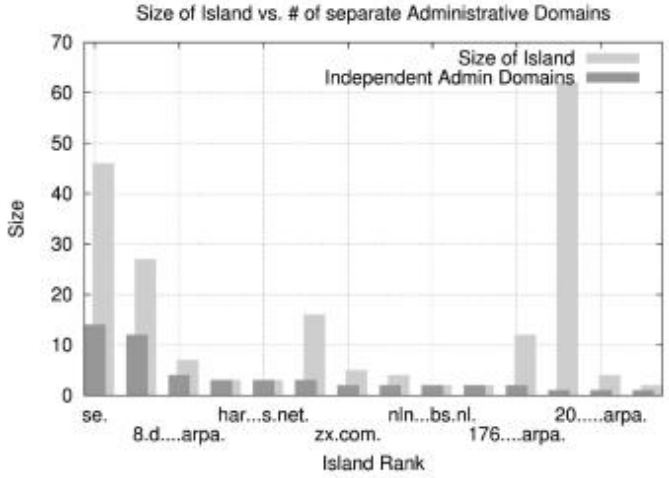

Fig. 5. This figure shows the size of several DNSSEC islands of security in contrast to the apparent number of independent administrative domains for each. Some islands of relatively large size appear to be run by a small number of independent administrative domains, while other islands are more diverse.

7 讨论

到目前为止，我们已经仔细研究了 DNSSEC 从简单的加密设计愿景到互联网上部署遇到的许多操作问题。这些主要由互联网的大规模和异构性引起的问题有时甚至使DNSSEC协议设计失效。

- 寻求统一对各个问题的分段分析，并分析在大规模系统中部署加密方案较为困难的根本原因
- 讨论分布式监控的重要性，并解决前面几节中提出的两个未决问题：
  - 增量部署和密钥撤销

讨论
7.1 为何如此困难?

As the saying goes, "In theory, there is no difference between theory and practice. But, in practice, there is" [33].

"理论上，理论与实践没有区别。但是，在实践中存在。"

- 除非可部署，否则密码学上的正确设计是没有用的
  - 要求：支持任意大小区域扩展、支持增量部署、与 DNS 缓存共存、容忍互联网中广泛存在的操作异构性
- 该系统跨越数千万个独立的管理域，DNSSEC 中定义的安全性的提供需要所有域的协同行动
  - 域公钥的任何本地更改都可能需要跨管理域同步，产生子区域较多的域拓展问题，协调过程漫长且易出错
  - 分层 PKI 给增量部署的挑战：
    - DNSSEC假设**从上到下**进行系统部署，若独立于其父区域进行配置，则可能造成PKI不完整，形成孤岛
- 缓存造成的独特挑战：
  - 区域公钥的更改不会立即对所有解析器可见，旧密钥将继续被整个网络上的缓存解析器使用，直到它们到期
  - 处理新旧密钥的共存期会导致设计复杂化和系统脆弱性
- 不同管理域中的区域可能以不同的做法进行操作（如对数据隐私保护的选择，离线/在线密钥存储）

讨论
7.2 分布式监控的重要性

- DNSSEC 监控工作暴露了许多仅在实际部署中出现的问题：
  1. 互联网中的**数据并不总可用**，瞬时故障、错误配置等现实问题需要被考虑
  2. **密码操作本身就是一个新的挑战**，缺乏基本的理解和经验导致了除了错误配置之外的额外错误
  3. DNSSEC 的设计假设与互联网各方倾向于自行决定是否/何时可以部署新功能或是否部署新功能的**共同要求不一致**
     - 出现大量孤立的 DNSSEC 孤岛，根本无法扩展
- 监控结果表明，**即使在早期部署阶段，DNSSEC也是一个高度动态且不断发展的系统**
  - 必须**持续监控**其行为以捕获新的故障和挑战
  - **为系统的设计提供信息、分析部署进度、发现问题并重新审视设计**

讨论
7.3 优化增量部署

在没有PKI的情况下，缓存解析器以可信的方式找到每个启用DNSSEC的区域的公钥

像SecSpider这样的分布式监控系统可以帮助实现这一需求：

- SecSpider已经提供了从所有已知DNSSEC启用区域收集的密钥
- **监控系统的分布式设计使得攻击者难以破坏结果**
- **不一致的回复将提供给解析器，让其自行决定**
- 增强信任的措施：
  1. **自我检查：**
     - 每个启用DNSSEC的区域可在SecSpider上注册
     - 定期检查其公钥的正确性
  2. **安全获取数据：**
     - SecSpider的公钥发布在其网站上
     - 通过多条查询路径获取公钥，防止中间人攻击

优点：

1. 利用公开性实现安全性：
- 数据公开，任何错误都暴露给受影响的区域
- 信任锚冲突可通过非公开渠道解决

2. 补充DNSSEC的PKI：

作为易于部署的补充解决方案，促进DNSSEC的渐进推出

讨论
7.4 解决密钥撤销的问题

针对私钥泄露时如何从缓存解析器中撤销旧密钥的问题，**本文作者认为应该将跟踪密钥状态的责任交给缓存解析器，并提出两种方案：**

1.定期密钥重新验证：

- 缓存解析器定期重新获取公钥
- 使用签名生成时间来判断密钥是否更新
- 删除由旧密钥签署的所有缓存数据

2. 按需数据重新获取：

- 缓存解析器检查是否有验证签名所需的公钥
- 若无，查询相应区域以获取缺失的公钥
- 重新获取数据和签名以保证验证
- 保证无论何时返回签名，它总有相应的公钥来验证签名

讨论
7.5 处理动态更新

- **动态更新记录需要在线密钥，但在线存储私钥有安全风险**
- **区域分割技术（混合方法）**
  - 将动态更新记录放在一个子区域（动态子区域）
  - 原始区域的私钥保持离线，动态子区域的私钥在线
- 步骤：
  1. 区分记录类型：手动 / 动态更新记录
  2. 创建动态子区域：
     - 将动态更新记录放入动态子区域
     - 例如：zonefoo.com 和 dyn.foo.com
  3. 使用CNAME动态更新：
     - 如 foo.com 及其动态子区域 dyn.foo.com，一个具有动态更新的A记录的主机 host.foo.com
     - host.foo.com CNAME host.dyn.foo.com
     - 实现动态更新

优点

- 结合在线和离线密钥的优点：
  - 动态子区域的在线密钥用于签署动态数据
  - 原始区域的私钥保持更高安全性
- 降低安全风险：
  - 在线密钥被破坏的风险最小化
  - 攻击者只能伪造动态子区域的记录，不能破坏原始区域
  - 防止对下级区域的级联损害

8 相关工作

自从[6]的开创性工作以来，DNS安全性引起了研究和运营社区的广泛关注。IETF成立了DNS扩展工作组（dnsext）以领导DNSSEC的发展，迄今发布了许多DNSSEC相关规范，尽管一些设计方面仍在发展，当前的DNSSEC规范集已被认为相对稳定和成熟[5][8][9][13][2][4][3][16][32][29][22]。

大多数DNSSEC规范关注的是已经做了什么，而不是为什么这么做，并且许多努力背后的理由和见解在相关公共资料极度缺失。

本文首次系统地记录了DNSSEC设计和部署问题，并在统一框架中对其进行分类；揭示了许多问题的相互关系及其根源，并提出了几种实用技术来促进DNSSEC的推广和运营。

虽然[28]指出安全系统在实施、可用性和应用集成方面面临挑战，但未通过具体案例进行分析。

与我们工作直接相关的是[20]对美国国防部大型**PKI部署**的研究，许多观察结果也适用于DNSSEC。特别是密钥撤销也是PKI部署中的最大技术挑战之一。

9 总结

Lesson1: Desgin for scalability 可拓展性的设计

Lesson2: Desgin for heterogeneity 异构设计

Lesson3: Desgin for incremental deployment 增量部署设计

Lesson4: Desgin for imperfect operations 针对不完美操作的设计

Lesson5: Desgin for monitoring as an integral component 将监控作为一个不可或缺的组件进行设计

附加参考资料

- Secspider https://secspider.net/
- DNSSEC分析器 https://dnssec-analyzer.verisignlabs.com/
- DNSSEC详解 https://zhuanlan.zhihu.com/p/355579999
- DNSSEC简介 https://www.guozeyu.com/2018/03/dnssec/
- 中科大DNSSEC https://free.eol.cn/edu_net/edudown/cernet2020/IPv6/zhj.pdf

Search:

	单位	网站	DNSSEC	v4 HTTP	v4 HTTPS	v4 HTTP2	v6解析	v6 HTTP	v6 HTTPS	v6 HTTP2	评分
1	网易	www.163.com		✔	✔	✔	✔	✔	✔	✔	90
2	携程旅行网	www.ctrip.com		✔	✔	✔	✔	✔	✔	✔	90
3	小米	www.mi.com		✔	✔	✔	✔	✔	✔	✔	90
4	新浪	www.sina.com.cn		✔	✔	✔	✔	✔	✔	✔	90
5	百度	www.baidu.com		✔	✔		✔	✔	✔		70
6	京东	www.jd.com		✔	✔		✔	✔	✔		70
7	美团点评	www.meituan.com		✔	✔		✔	✔	✔		70
8	腾讯	www.qq.com		✔	✔		✔	✔	✔		70
9	搜狐	www.sohu.com		✔	✔		✔	✔	✔		70
10	阿里巴巴	www.alibaba.com		✔	✔	✔					50
11	三六零	www.360.cn		✔	✔						40

Showing 1 to 11 of 11 entries

	单位	网站	DNSSEC	v4 HTTP	v4 HTTPS	v4 HTTP2	v6 解析	v6 HTTP	v6 HTTPS	v6 HTTP2	评分
1	UPenn	www.upenn.edu	✔	✔	✔	✔	✔	✔	✔	✔	129
2	普林斯顿	www.princeton.edu	✔	✔	✔	✔	✔	✔	✔	✔	126
3	斯坦福	www.stanford.edu	✔	✔	✔	✔	✔	✔	✔	✔	125
4	剑桥	www.cam.ac.uk	✔	✔	✔	✔	✔	✔	✔	✔	124
5	UCL	www.ucl.ac.uk	✔	✔	✔	✔	✔	✔	✔	✔	113
6	Caltech	www.caltech.edu	✔	✔	✔	✔	✔	✔	✔	✔	110
7	ETH	www.ethz.ch	✔	✔	✔	✔	✔	✔	✔	✔	106
8	UChicago	www.uchicago.edu	✔	✔	✔	✔	✔	✔	✔	✔	105
9	Umich	umich.edu	✔	✔	✔		✔	✔	✔		80
10	IC	www.imperial.ac.uk	✔	✔	✔		✔	✔	✔		80
11	NUS	www.nus.edu.sg	✔	✔	✔	✔					60
12	牛津	www.ox.ac.uk	✔	✔	✔	✔					60
13	Cornell	www.cornell.edu		✔	✔	✔	✔	✔	✔	✔	90
14	爱丁堡大学	www.ed.ac.uk		✔	✔	✔	✔	✔	✔	✔	90

	单位	网站	DNSSEC	v4 HTTP	v4 HTTPS	v4 HTTP2	v6 解析	v6 HTTP	v6 HTTPS	v6 HTTP2	评分
1	中国科学技术大学	www.ustc.edu.cn	✔	✔	✔	✔	✔	✔	✔	✔	139
2	中国农业大学	www.cau.edu.cn	✔	✔	✔		✔	✔	✔		80
3	东北大学	www.neu.edu.cn	✔	✔	✔		✔	✔	✔		80
4	上海交通大学	www.sjtu.edu.cn	✔	✔	✔		✔	✔	✔		80
5	重庆大学	www.cqu.edu.cn		✔	✔	✔	✔	✔	✔	✔	90
6	西北农林科技大学	www.nwsuaf.edu.cn		✔	✔	✔	✔	✔	✔	✔	90
7	中国人民大学	www.ruc.edu.cn		✔	✔	✔	✔	✔	✔	✔	90
8	清华大学	www.tsinghua.edu.cn		✔	✔	✔	✔	✔	✔	✔	90
9	武汉大学	www.whu.edu.cn		✔	✔	✔	✔	✔	✔	✔	90
10	厦门大学	www.xmu.edu.cn		✔	✔	✔	✔	✔	✔	✔	90
11	浙江大学	www.zju.edu.cn		✔	✔	✔	✔	✔	✔	✔	90
12	北京航空航天大学	www.buaa.edu.cn		✔	✔	✔	✔	✔	✔		80
13	湖南大学	www.hnu.edu.cn		✔	✔		✔	✔	✔	✔	80

谢谢！