松：启用隐私保护的 TLS 深度包检测与基于规则隐藏和快速连接建立

第·章 2020 年 9 月

DOI: 10.1007 / 978-3-030-58951-6_1

Pine：
通过
规则隐藏和快速连接
建立在 TLS 上保护隐私的深度数据包检查

尖挺宁[1,4]，新艺黄[1(乙)]，GEONG 森婆[2]，生民徐[1]，
加 Chng（更改）洛[2]，坚翁[3]，和 Robert H.邓[4]

[1] 福建师范大学数学与信息科学学院福建省网络安全与密码学重点实验室，福州
jtning88@gmail.com， xyhuang81@gmail.com， smxu1989@gmail.com
[2] 新加坡国大新加坡网络安全实验室，新加坡
pohgs@comp.nus.edu.sg， dcsljc@nus.edu.sg
[3] 暨南大学信息科学与技术学院，中国广州
cryptjweng@gmail.com
[4] 新加坡管理大学信息系统学院，新加坡，新加坡

抽象的。 传输层安全检查（TLSI）使企业能够在将用户流量路由到目的地之前对其进行解密，检查和重新加密。这破坏了 TLS 规范和实施的端到端安全保证。由于企业的用户现在已经知道用户的流量，因此这也引起了隐私方面的顾虑，并且提供检查服务的第三方中间盒提供商可能会另外了解企业的检查或攻击规则，策略。BlindBox（SIGCOMM 2015）和 PrivDPI（CCS 2019）这两项最新著作提出了隐私保护方法，可以直接检查加密流量，以解决用户流量的隐私问题。但是，在 TLS 连接建立期间，BlindBox 会产生较高的预处理开销，尽管 PrivDPI 大大减少了开销，但与 TLSI 相比仍然值得注意。此外，两种方法的基本假设是中间盒知道规则集。但是，随着服务越来越多地迁移到基于第三方云的设置，应该保留规则隐私。同样，从添加任何规则都需要大量预处理和重新实例化的意义上来说，这两种方法本质上都是静态的。

在本文中，我们提出了 Pine，一种新的加密流量协议的隐私保护检查，该检查（1）简化了 PrivDPI 的预处理步骤，从而进一步减少了在用户与服务器之间建立 TLS 连接的计算时间和通信开销；（2）支持*规则隐藏*；（3）启用动态规则添加，而无需从头开始重新执行协议。我们展示了
L. Chen 等。（编辑）：ESORICS 2020，LNCS 12308，第 3-22 页，2020. https://doi.org/10.1007/978-3-030-58951-6 _ 1

通过广泛的实验，与 PrivDPI 相比，Pine 具有更出色的性能。特别是，对于具有 5,000 个令牌和 6,000 个规则的从客户端到服务器的连接，Pine 的速度提高了大约 27%，并节省了大约 92.3%的通信成本。

关键字： 网络隐私.流量检查.加密流量

1 引言

根据最近的互联网趋势报告[11]，今天的 Web 流量中有 87%是经过加密的，而 2016 年为 53%。同样，跨 Google 的 94%以上的 Web 流量都使用 HTTPS 加密[7]。越来越多地使用端到端加密来保护 Web 流量，这妨碍了现有中间盒通过对流量进行深度数据包检查来检测恶意数据包的能力。结果，安全服务提供商和企业部署了执行中间人（MitM）的工具，以在将流量发送到指定服务器之前对流量进行解密，检查和重新加密。这种方法被国家安全局（NSA）称为传输层安全检查（TLSI），该机构最近发布了有关 TLSI 的咨询[12]引用了包括内部威胁在内的潜在安全问题。TLSI 带来了其他风险，管理员可能会滥用其权限来从解密的流量中获取敏感信息。另一方面对中间盒和企业网关对用户数据的访问的隐私问题越来越多。根据最近对美国 TLSI 的调查[16]，超过 70%的参与者担心执行 TLSI 的中间盒（或 TLS 代理）可能被黑客利用或被政府使用，近 50%的参与者认为这是对隐私的入侵。通常，参与者可以接受雇主或大学出于安全目的使用中间盒的要求，但还希望确保不会将这些中间盒用于政府监视或被黑客利用。

为了减轻上述对维护 TLS 安全性的担忧，同时又确保了加密流量的私密性，Sherry 等人。 [20]引入了一个名为 BlindBox 的解决方案，可以直接对加密流量进行检查。但是，BlindBox 需要在中间盒和客户端之间执行的设置阶段。设置阶段执行两方计算，其中中间盒的输入是规则，这意味着无法确保针对中间盒的规则的私密性。此外，此设置阶段是基于乱码电路构建的，需要为每个会话执行。由于乱码电路的特性，这种设置阶段会导致大量的计算和通信开销。为了克服这个限制，Ning 等 人。 [15]最近提出了具有改进设置阶段的 PrivDPI。引入了一种新的模糊规则生成技术，该技术可以跨后续会话重用在第一个 TLS 会话期间生成的中间值。这大大减少了一系列会话的计算和通信开销。但是，由于需要每个客户端为每个新连接运行预处理协议，因此在建立 TLS 连接期间仍然存在相当大的延迟。此外，

正如我们将在 Sect 中显示的那样。4.1 ，当检查或攻击规则的范围较小时，中间盒可以在 PrivDPI 设置中对规则进行暴力猜测。这意味着，与在 BlindBox 中一样，PrivDPI 不提供针对中间盒的规则保密性。但是，如 [20]中所述，大多数解决方案提供商（例如 McAfee）都依赖其业务模型中规则的隐私性。鉴于基于云的中间盒服务越来越流行，更应该针对中间盒保留规则的私密性。

鉴于 TLSI 的安全性和隐私问题以及最新技术的现状，我们除了维持 BlindBox 和 PrivDPI的安全性和隐私条款外，还寻求引入一种新的解决方案来解决以下问题：*（1）无需预处理即可快速建立 TLS 连接，以消除 BlindBox 和 PrivDPI 所引起的会话建立延迟；（2）即使对于较小的规则域，也可以抵抗规则集的蛮力猜测；（3）支持轻量级规则添加。*

我们的贡献。我们为实际的企业网络设置提出了一种新的协议 Pine，该协议用于对加密流量进行隐私保护的深度包检查，以保护客户端通过企业网关连接到 Internet。主要贡献概述如下。

- **确定 PrivDPI 的局限性。**我们将重新审视 PrivDPI，并演示在 PrivDPI 中，当规则域较小时，中间盒可以伪造新的加密规则，从而使中间盒能够使用其生成的任何加密规则来检测加密流量。
- **具有更强隐私保护的新解决方案。**我们建议使用 Pine 作为解决保留隐私的深度数据包检查问题的新解决方案，以确保更强的隐私性。首先，除非流量中存在攻击，否则流量的私密性得到保护。此外，针对中间盒确保了规则的私密性，我们将此属性*规则*称为*隐藏*。即使规则域很小（例如，在现有的网络入侵检测（IDS）规则中大约为 3000 条规则），此属性也可以确保规则的私密性，从而解决了 PrivDPI 的局限性。另外，还针对企业网关和端点确保了*规则的隐私*性，我们将其称为“属性*规则隐私*”。
- **摊销设置，快速建立连接。**Pine 可以建立具有低延迟的 TLS 连接，并且不需要像 PrivDPI 和 BlindBox 中那样的交互式预处理协议。引起延迟的预处理协议是离线执行的，仅执行一次。因此，没有每个用户连接的开销。任何客户端都可以与远程服务器建立安全的 TLS 连接，而无需进行预处理延迟。相反，在 PrivDPI 和 BlindBox 中，规则越多，每个连接的安装成本就越高。连接速度对于低延迟应用程序至关重要。
- **轻量级规则添加**。Pine 是一种动态协议，它允许动态添加新规则，而不会影响客户端和服务器之间的连接。在网关可以通过中间盒本地执行规则添加阶段的意义上规则添加对客户端是无缝的

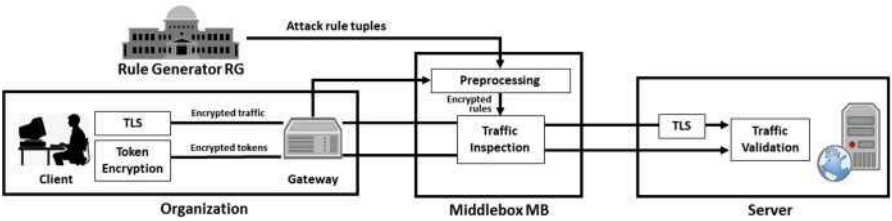

图 1. Pine 系统架构。

无需任何客户参与。与 BlindBox 和 PrivDPI 相比，这是有益的，在 BlindBox 和 PrivDPI 中，客户端需要为每个连接从头开始重新运行预处理协议。

除了提供更强的隐私保护外，我们还进行了广泛的实验，以证明 Pine 与 PrivDPI 相比具有优越的性能。对于从客户端到具有 5,000 个令牌和 6,000 个规则集的服务器的连接，Pine 比 PrivDPI 快 27%，并节省大约 92.3%的通信成本。特别是，Pine 的通信成本与规则数量无关，而 PrivDPI 的通信成本与规则数量成线性增长。

2 协议概述

如图 1 所示，Pine 与 BlindBox 和 PrivDPI 共享相似的体系结构。Pine 中有五个实体：客户端，服务器，网关（**GW**），规则生成器（**RG**）和中间盒（**MB**）。客户端和服务器是发送和接收受 TLS 保护的网络流量的端点。**GW** 是位于一组客户端和服务器之间的设备，它允许网络流量从一个端点流向另一个端点。**RG** 为 **MB** 生成攻击规则元组。**MB** 将使用攻击规则元组检测网络流量中的攻击。每个攻击规则都描述了一次攻击，并包含一个或多个在网络通信中要匹配的关键字。此后，我们将互换使用术语“规则”和“攻击规则”。**RG** 的角色可以由诸如 McAfee [18]的组织来执行。**MB** 是使用 **RG** 发出的攻击规则元组检查和过滤网络流量的网络设备。

系统要求。主要目的是提供一种隐私保护机制，该机制可以检测任何可疑流量，同时确保端点流量的隐私。特别是，系统要求包括：

- *流量检查*：Pine 保留了传统 IDS 的相似功能，即在数据包中查找可疑关键字。
- *规则隐私*：端点和 **GW** 不应学习攻击规则（即关键字）。对于生成综合和专有规则集作为其独特命题的安全解决方案提供商而言，这是特别必要的，有助于更有效地检测恶意流量。

- *流量隐私*：一方面，除了符合规则的流量部分之外，**MB** 不应学习网络流量的明文。另一方面，不允许 **GW** 读取流量的内容。
- *规则隐藏*：**MB** 是不是应该学习从发出攻击规则的元组攻击规则 **RG** 在基于云的环境 **MB** 驻留在云平台上。在这种情况下，无法完全信任基于云的中间盒。安全解决方案提供商可能希望保护其唯一规则集的隐私，如前面在描述规则隐私时所述

威胁模型。攻击者分为以下三种类型。

- *恶意端点*。第一类攻击者是端点（即客户端或服务器）。与 BlindBox [20]和 PrivDPI [15]相似，两个端点中的最多一个被认为是恶意的，但不是两个。这种攻击者与传统 IDS 中的攻击者相同，其主要目标是逃避检测。与传统的 IDS [17] 一样，这是两个端点中至少一个诚实的基本要求。这是因为，如果两个恶意终结点在私钥上达成一致并发送使用此特定密钥加密的流量，那么检测恶意流量将是不可行的。
- *网关的攻击者*。与常规网络设置一样，**GW** 被认为是半诚实的。也就是说，**GW** 诚实地遵循协议规范，但可以尝试学习流量的明文。**GW** 也可能尝试从接收到的消息中推断规则。
- *中间盒的攻击者*。**MB** 被认为是半诚实的，它遵循该协议，但可能会尝试从接收到的消息中学习超出允许范围的信息。特别是，它可能会尝试读取通过它的流量的内容。此外，它可能会尝试学习 **RG** 发出的攻击规则元组的基本规则。

协议流。我们介绍了各个阶段的功能，如下所示。

- 初始化。**RG** 通过设置公共参数来初始化系统。
- 设置。**GW** 订阅了 **RG** 的检查服务，其中 **RG** 接收到 **GW** 的共享机密。**RG** 向 **MB** 发布攻击规则元组。客户端和服务器将从主要 TLS 握手协议的密钥派生一些参数并分别安装 Pine HTTPS 配置。
- 预处理。在此阶段，**GW** 与 **MB** 交互以生成一组可重用的随机规则。另外，**GW** 生成初始化参数并将其发送到其域内的客户端。
- 会话检测规则的准备。在此阶段，可重用的随机规则将用于生成会话检测规则。
- 令牌加密。在此阶段，客户端为有效负载中的每个令牌生成加密令牌。加密的令牌将与使用常规 TLS 从有效负载加密的流量一起发送。
- 网关检查。对于第一个会话，**GW** 检查客户端发送的附加参数格式是否正确。当客户端首次连接到服务器时，将运行此阶段。

- 交通检查。**MB 会**生成一组加密规则，并使用这些加密规则执行检查。
- 交通验证。如果另一个端点是恶意的，则一个端点执行流量验证。
- 规则添加。在此阶段，将添加一组新的攻击规则。**GW** 与 **MB** 交互以生成与这些新攻击规则相对应的可重复使用的随机规则集。

3 初赛

复杂性假设。决策 Diffie-Hellman（DDH）问题描述如下：给定 g，g^x，g^y，g^z，确定 $z = xy$（模为 g 的阶），其中 x，y，z e Z ₚ。我们说，如果| | ，则 PPT 算法 B 在解决 DDH 问题中具有优势 e。Pr [B ⁽ˢ· ᵍ⁼ ⁾ = 1]-Pr [B ˈ s'A “ A ， g^z ⁾ = 1] | > ë， 上面的概率接管了 B，g， x，y，z 的硬币。

定义 1.*如果没有 PPT 对手在解决 DDH 问题方面至少具有 e 优势，则 DDH 假设成立。*

伪随机函数。伪随机函数族 PRF 是一个家庭的函数{ PRF –个：*UV* |一个 *ê* A}，使得 *甲* 可能是有效 samplable 和所有 PRF ，U，V，A 是由一个安全参数 A.索引的所述的安全属性 PRF 是：对于在 A 中运行的任何 PPT 算法 B，它认为| Pr [B ^{PRF} 。⁽ ∧ ⁾ = 1] — Pr [B ᴿ⁽ ∧ ⁾ = 1] | = negl（A），其中 negl 是 A，a 和 R 的可忽略函数 分别在 A 和（UV）上均匀。上面的概率接管了 B，a 和 R 的硬币。为了简化符号，我们考虑了通用伪随机函数概念的一种版本，该概念是为适应我们的实现而定制的。具体来说，本文考虑的伪随机函数 PRF 将 A 位字符串映射到 Z ₚ的元素。即，PRF ₐ：{0,1} ᴬ Z ₚ，其中 a *e* *G*。

有效负载令牌化。像在 BlindBox 和 PrivDPI 中一样，我们部署基于窗口的令牌化来令牌化客户端有效负载的关键字。基于窗口的标记化遵循简单的滑动窗口算法。实施协议时，每个令牌采用 8 个字节。也就是说，给定有效载荷“密钥”，端点将生成令牌“密钥 k”，“密钥 ke”和“密钥”。

4 协议

在本节中，我们首先指出 PrivDPI 的局限性。为了解决此问题并进一步减少连接延迟我们然后提出了新协议。

4.1 PrivDPI 的局限性

我们展示了当规则域较小时 PrivDPI 如何失败。我们说，如果可以通过蛮力攻击来猜测给定公共参数的基本规则，则规则的范围很小。我们首先回顾一下

PrivDPI。在设置阶段，中间盒收到 规则 r i的sig （R i）），其中[R 我=克 AR 我+ $ 我和 SIG （R 我）是的签名 ř 我。带 S ĵ 和 R 我，**MB** 获得值 克 AR 我。回想一下，在 PrivDPI 中，值 $A = g^{a}$ 包含在 PrivDPI HTTPS 配置中，**MB** 可以通过安装 PrivDPI HTTPS 配置来获取该值。由于规则的域很小，因此 A 和 g^{ar*}，**MB** 可以尝试通过进行蛮力攻击来获得 r_i 的值吗？

通过在规则域内检查 $A^{v} = g^{ar_{i}}$ 候选值 *v*。以这种方式，**MB** 可能获得值 r - [R 我和- [R ĵ为 ř ĵ。预处理协议完成后，**MB** 获得规则 r i的可重复使用的模糊规则 $I_i = g^{kar_j + k2}$。

现在，**MB** 知道值 r j，r j，$I_i = g^{kar_i + k}$，$I_j = g^{kar_j + k}$。然后，它可以计算（I 我/ I ĵ ）^{R 我 -[R j），以获得值 克 KA。有了 g^{ka}，r_i 和 $I = g^{kar_j + k}$，就可以计算出 $I_i / (g^{ka})^{r_j} = g^{k}$。与 g^{k} 和 g^{ka}，**MB** 可以针对它选择的任何规则成功地伪造可重复使用的模糊规则。使用伪造（但有效）的可重复使用的模糊规则，**MB** 可以检测到超出其允许范围的规则，这违反了加密流量的隐私要求。

4.2 我们的协议说明

初始化。令 R 为规则域，PRF 为伪随机函数，*n* 为规则数，[n]为集合{1，*n}*。设 AES a（salt ）为密钥 *a* 和消息 salt 的 AES 加密。令 Enc a（salt ）= AES a（salt ）mod R ，其中 R 是用于减小密文大小的整数[20]。初始化阶段接受安全参数 A 并选择组 *G* 素数 *p*。然后，它选择 G 的生成器 *g*，并将公共参数设置为（*G*，*p*，*g*）。

设置。GW 为伪随机函数 PRF 选择密钥 g^{w}，其中 *w* G Z *。它从 **RG** 订阅服务并将 w 发送到 **RG**。**RG** 首先计算 $W = g^{w}$。对于规则集{r GR}即[n]，对于 *i* G [n]，**RG** 选择一个随机性 k_i *G* Z p，计算 $r_{w,i} = \mathrm{PRF}_{w}$（r i）且 $R_i = g^{r_{w,i} + k_i}$。**RG** 选择一个签名方案，其中 *sk* 为私钥，*pk* 为公钥。然后，它用 *sk* 签名{R j } ^^]，并 为 *i G* [n]生成 R_i 的签名，用 a i表示。最后，它发送*攻击规则元组{（R i，ff i，ki i）} iE [n]转换*为 **MB**。在此，g^{w} 是确保规则隐藏属性的关键要素。这里关键的一点是，由于 **MB** 不知道 $g^{\wedge w \wedge}$ 或 W，它不能推测底层[R 我的[R 我通过暴力破解所有它选择可能的关键字。特别是，对于给定的攻击规则的元组（R 我，置于<J 我，K 我），**MB** 可以得到的值 克 f 我和k 我通过计算 *RI*，/ 克 k 我。由于伪随机函数的性质，r wji 是伪随机的，因此是伪随机的。在不知道 g^{w} 或 w 的情况下，即使 **MB** 蛮力强制选择所有可能的关键字，也无法获得 r_i。

另一方面，客户端和服务器将安装一个包含值 R 的 Pine HTTPS 配置。令 k_{sk} 为客户端和服务器建立的常规 TLS 握手协议的密钥。与 k_{SK}，客户端（相应的服务器）导出三个键 k_{T}，C，$k_{小号}$。具体来说，k_T 是标准的 TLS 密钥，用于加密流量。*c* 是来自 Z p的随机值，用于生成会话检测规则；k_s 是来自 Z p的随机值，用作屏蔽从客户端发送到服务器的参数的随机性。

预处理。为了加速客户端和服务器之间的网络连接（与 PrivDPI 相比），我们引入了一种新方法，该方法可以快速建立连接，而无需像在 PrivDPI 中那样对每个客户端执行预处理过程。我们从企业环境中的常见网络场景开始，在企业环境中，在一组客户端和服务器之间存在一个网关。主要思想是让网关成为其域内客户端的代表，这些客户端将使用 **MB** 运行预处理协议仅一次。客户端和网关都共享与服务器连接所需的初始化参数。在这种情况下，由于预先由网关和 **MB** 执行预处理，因此无需像 PrivDPI 中那样进行任何预处理就可以立即建立客户端和服务器之间的连接。换句话说，我们将预处理操作转移到网关，这大大减少了客户端和服务器之间连接的计算和通信开销。

具体而言，在此阶段，**GW** 运行带有 **MB** 的预处理协议以生成*可重用的随机规则*集以及 **GW** 域内客户端的初始化参数。预处理协议在 TLS 握手协议之后运行，该协议在图 2 中进行了描述。完成此阶段后，**MB 将**获得一组可重用的随机规则，这些规则使 **MB** 能够跨一系列会话对加密流量执行深度数据包检测。值 $我_0$，并且 $余_2$ 启用的域内的每个客户端 **GW** 生成加密令牌。因此，与 BlindBox 和 PrivDPI 相比，对于与服务器的任何网络连接，客户端都不需要使用 **MB** 运行预处理阶段。这大大减少了客户端和服务器之间的网络连接的延迟和通信成本，尤其是对于大型规则集。此外，在添加新规则的情况下，客户端不需要像 BlindBox 和 PrivDPI 一样重新运行预处理协议。这意味着添加规则对客户端没有影响。

会话检测规则的准备。在此阶段将生成一组会话检测规则。这些会话检测规则是根据预处理协议生成的可重复使用的随机规则为每个会话量身定制的。生成的会话检测规则用作生成相应加密规则的输入。该协议在图 3 中进行了描述，并针对每个新会话执行该协议。

令牌加密。与 BlindBox 和 PrivDPI 相似，我们采用了 Sect 中介绍的基于窗口的标记化方法。3。在令牌化步骤之后，客户端获得与有效载荷相对应的一组令牌。客户端第一次与服务器连接时，客户端会 从 c 导出一个盐并存储该盐以备将来使用，其中 c 是从 TLS 握手协议的密钥 k_{sk} 派生的密钥。对于每个令牌 t，客户端运行如图 4 所示的令牌加密算法。为了防止伯爵表 牛逼变得过大，则客户端将清除牛逼 每 $ž$ 会话（例如，$ž$= 1,000）。在这种情况下，客户端会将新的盐发送到 **MB**，其中盐 $\text{salt} + \max_t \text{count}_t + 1$。

在上面，我们描述了当端点是客户端时的令牌加密。当端点是服务器时，服务器将首先运行相同的令牌化步骤，并按照图 4 中所述的步骤 1 和步骤 2 对令牌进行加密。

网关检查。当客户端首次连接到服务器时，将执行此阶段。对于第一次从客户端发送到服务器的流量，客户端将附加（salt，C_{ks}，C_w，C_x，C_y）。这使服务器能够在流量验证阶段执行加密流量的验证。C_{ks}和 k_s作为掩盖值 g^w，g^x和 g^{xy}的随机性。流量到达服务器后，将检查 C_{ks}的正确性。为了确保 g^w，g^x和 g^{xy} 被 C_{ks}正确屏蔽，**GW** 仅检查以下等式是否成立：C_w=（C_{ks}）^{w}，C_x=（C_{ks}）^{x}和 C_y=（C_{ks}）^{xy}。

交通检测。在流量检测阶段，**MB** 在流量中的加密令牌与其保留的加密规则之间执行相等性检查。流量检测算法描述如下。**MB** 首先初始化一个计数器表 CT_{r_i} 以记录每个规则 r_i的加密规则 E_{r_i}。规则的加密规则 E_{r_i}计算为 E_{r_i} = Enc_{s_i}（salt + $count_{r_i}$），其中 $count_{r_i}$初始化为 **0**。**MB** 随后生成包含加密规则的搜索树。如果找到匹配项，**MB** 将采取相应的措施，删除 与 n 对应的旧 E_{r_i}，将 $_{r_i}$**计数**增加 1，计算并将新的 E_{r_i}插入树中，其中新的 E_{r_i}计算为 Enc_{s_i}（**盐+计数** $_{r_i}$）。

流量验证。如果这是客户端与服务器之间的第一个会话，则在接收到（salt，C_{ks}，C_w，C_x，C_y）后，服务器将检查方程 C_{ks} = g^{k_s}成立，从而得出 k_s（由服务器）从常规 TLS 握手协议的密钥 k_{sk}获取。如果等式成立，则服务器计算（C_w）$^{(k_s)}$ = g^w，（C_x）$^{(k_{s^{-1}})}$ = g^x，（C_y）$^{(k_s)^{-1}}$ = g^{xy}。使用计算出的（g^w，g^x，g^{xy}），服务器对从服务器解密的明文运行相同的令牌加密算法。

输入： **MB** 具有输入 $\{\{R_i, \wedge_i, k_i\}\}_{i\in[1..]}$，其中 $R_i = g^{xw, i+ki}$；**GW** 已输入 *pk*。该协议在 **GW** 和 **MB** 之间运行：

1. **GW** 选择一个随机 $x \in Z_p$，计算 $X = g^x$，然后将 X 发送给 **MB**。
2. **MB** 发送{（[$R_{我}$，一个$_{我}$）}$_{我\in[n]}$到 **GW**。
3. 在收到{（R_i，a_i）}$_{i\in[n]}$时，**GW** 执行以下操作：
   (1) 使用 pk 表示 $i \in [n]$，检查 c_i是否是 R_i上的有效签名；如果不是，则停止并输出±。
   (2) 选择一个随机 $y \in_{1*p}$并计算 $Y = g^y$。对 $i \in [n]$计算 X_i = $(R_i \bullet Y)^x = g^{xr \wedge\wedge i + xki + xy}$，然后将{$X_i$}$_{i\in[n]}$返回给 **MB**。
4. **MB** 计算 $k_{我}$= $X_{我}$/（X）$^{=}$克 $^{xRw, 我}$ ^{xy}用于我 ǵ [Ñ]作为规则可重复使用的随机化的规则-$[R_{我}$。
5. **GW** 将 I_o= xy，I_1 = x，$I_2 = g^{w设置}$为初始化参数，并将（I_0，I_1，I_2）发送到其域内的客户端。

图 2.预处理协议

*输入：*客户端（即服务器）具有输入 c。**MB** 已输入$\{K_i\}_{i\in[n]}$。
该协议在客户端，服务器和 **MB** 之间运行：

1. 客户端计算 $C = g^c$ 并将 C 发送到 **MB**（通过 **GW**）。同时，服务器设置 $C_s = c$ 并将 C_s 发送到 **MB**。
2. **MB** 检查 C 是否等于 g^{C_s}。如果是，则对于 $i \in [n]$，它计算 $S_i = (K_i \cdot C)^{C_s} = g^{c(x_i w_i + x_i y + c)}$ 作为规则 r_i 的会话检测规则。

图 3.会话检测规则准备协议

像客户端一样加密 TLS 流量。然后，服务器检查生成的加密令牌是否等于从 **MB** 接收的加密令牌。如果不是，则表明客户端是恶意的。另一方面，如果这是从服务器发送到客户端的流量，则客户端将执行与服务器相同的令牌加密算法，并将获得的加密令牌与从 **MB** 接收到的加密令牌进行比较。

规则添加。实际上，可能需要将新规则添加到系统中。对于一个新的规则- *[R 我* GR 为*我* G [N ']， **RG** 随机选择 *ķ 我* ġ $\check{Z}_p$，算出 R'= PRF（*R 我*）和- *[R 我*= 。 然 后对生成的 *R i* 进行签名

与 SK 生成签名一个*我*的- *[R 我*。最后，它将新添加的攻击规则元组*{ Ri，a i，k i* 发送给 **MB**。对于新添加的攻击规则元组，在图 5 中描述了规则添加协议，该协议是预处理协议的简化协议。

*输入：*客户端具有输入（I_0，I_1，I_2），令牌 *t*，随机密钥 k_s 和 c，值 R，盐和计数器表 T，其中 $I_0 = xy$，$I_1 = x$ 和 $I_2 = g^w$。
该算法由客户端运行，如下所示：

1. 计算 $I = I_0 + c = xy + c$。
2. 对于每个令牌 *t*：
   - 如果在 T 中不存在与 t 对应的元组：计算 t_w = PRF（t），$T_t = g^{c(I_1 t_w + I)} = g^{c(x t_w + xy + c)}$，设置计数$_t$ = 0，将 t 的加密计算为 E_t = Enc$_{T_t}$（盐）。最后，将元组（*t*，T_t，count$_t$）插入到 T 中。
   - 如果存在一个元组（吨'T_t，县中）Ť 其中 吨' =吨：更新县=县+ 1，并且计算的加密吨作为 ë=的 Enc（盐+县）。
3. 如果是第一次会话，则计算 $C_{ks} = g^{ks}$，$C_w = (I_2)^{ks} = g^{wks}$，$C_x = g^{I_1 ks} = g^{xks}$ 和 $C_y = g^{I_0 ks} = g^{xyks}$。参数（salt，$C_{ks}$，$C_w$，$C_x$，$C_y$）将与加密令牌一起发送 E_t 代表令牌 *t*。

图 4.令牌加密算法

*输入：***MB** 已新添加的攻击规则的元组集合*{（R 我，一个我，K 我）}*我è[Ñ"Ĵ，其中*[R 我= G"瓦特，我+ *我;* **GW** 具有输入 *Y，x。*
该协议在 **GW** 和 **MB** 之间运行：
1. **MB** 发送{（[R 我，一个我）}我è[Ñ'Ĵ到 **GW**。
2. 在接收到{（R."-.A"）}我è[Ñ"Ĵ，**GW** 的作用：（1）检查一个我是有效的签名 ř 我使用 *PK* 为*我* ġ [Ñ"]; 如果不是，则停止并输出±。（2）计算 *X 我*=（[R 我• Y）x =克 XR W ^ + XY 对于我克[Ñ']和发送{ X '}我è[Ñ/]到 **MB**。
3. **MB** 计算可重复使用的随机化的规则 *k 我*= X "-/（X）*用于我 ά[Ñ"]。

图 5.规则添加协议

5 安全性

5.1 中间盒可搜索加密

定义。对于消息空间 M，中间盒可搜索加密方案由以下算法组成：

- **设置**（A）：获取安全参数 A，输出密钥 sk 。
- TokenEnc （吨我， SK ）：取令牌集合{吨 J ë中号} J£[Ñ]和密钥 SK ，出放入一组密文（c 1，...， c n）和一个**盐**。
- RuleEnc （R，SK ）：取的规则*重新男，*密钥 SK ，输出加密的规则 Ë ř。
- **匹配**（ë [R，（c ^ 1，...， Ç Ñ），**盐**）：取加密的规则 ë [R，密文{ Ç 我}我£[Ñ]和**盐**，输出该组索引{ IND J } j£[l]，其中 ind j *e* [n]表示 *ie* [l]。

正确性。我们为读者提供附录 A 的定义。

安全。它在挑战者 *C* 和对手 A 之间定义。

- **设置。**C 运行**安装程序**（A）并获取密钥 sk 。
- **挑战。**A 从 *M 中*随机选择两组令牌 *S o* = {t o，i，...，t 0，n}，*S 1* = {t i，i，...，t i，n}并给出两组设置为 C。收到 S o和 S 1时，C 翻转随机硬币 *b，*运行 TokenEnc （t b，1，...，*t bn*， sk ）以获取一组密文（c 1，...， c n）和**盐**。然后将（c 1，...， c n）和**盐**赋给 A。
- **查询。**甲随机选择的一组规则（R 1，...，*R 米）*不同于 M 和给出的规则 C.当接收到一组规则，对于 i *ë*[米]，C 运行 RuleEnc （R J，SK ）获得加密的规则 *e r i*。C 然后将加密规则*{e r i }*即[TO]交给 A。
- **猜猜。**A 输出 *b* 的猜测 *b'*。

设 *I 0 j* 是与 S o中的 r 匹配的索引集，而 *I 1，j* 是与 S 1中的 r 匹配的索引集。如果 *I 0 i* = *I 1 j* 并且 *b '* = b 对于所有 i，我们说对手赢得了上述比赛。游戏中对手的优势被定义为 Pr [b ' = b]-1/2。

建造。下面的构造从安全的角度捕获了主要结构。

- **设置**（A）：令 PRF 为伪随机函数。生成 $x, y, c, w \in Z_p$，将（x，y，c，g^w）设置为键。
- TokenEnc（T_1，...，牛逼 $_n$，SK）：咱们**盐**是一个随机的盐。对于 $i \in [n]$，执行以下操作：（a）设 count 为令牌 A 在序列 t_1，...，t_{i-1} 中重复的次数；（b）计算 t_{wi} = PRF $_z$（t j），$T_{t_i} = g^{c(xw_i + xy + c)}$，Q = H（T_{t_i}，**盐+数**）。最后，算法输出（c_1，...，c_n）和 salt。
- RuleEnc（R，SK）：计算 $\tilde{r}_{瓦特}$= PRF $_{克瓦特}$（R），小号=克 $^{c(XX_{瓦特}+XY+c)}$ 中，输出 H（S）。

定理 1.假设 *H* 是一个随机预言，即 Sect 中的构造。5.1 是一种安全的中间盒可搜索加密方案。

该定理的证明在附录 B.1 中提供。

5.2 预处理协议

定义。预处理协议是 **GW** 和 **MB** 之间的两方计算。令 f：$\{0,1\}^* \times \{0,1\}^* \{0,1\}^* \times \{0,1\}^*$ 是计算过程，其中对于每个输入（*a*，b），输出为（f_1（*a*，*b*），f_2（*a*，*b*））。在我们的协议中，**GW** 的输入 是 *x*，而 **MB** 的输入是 *r* 的派生，只有 **MB** 接收到输出。

安全。安全要求包括：（a）**GW** 不应该学习每条规则的价值；（b）**MB** 无法伪造与在预处理协议期间获得的可重用随机规则不同的任何新的可重用随机规则。直观地，如果 **MB** 无法获得值 x，则满足第二个要求。由于 **GW** 和 **MB** 都被假定为半诚实的，因此我们采用具有静态半诚实对手的安全性定义，如[6]所示。假设 n 是用于计算 f 的两方协议，View $^\wedge$是执行 n 期间的第 i 方视图，而 Output ^{n} 是执行 n 时 **GW** 和 **MB** 的联合输出。对于我们的协议，由于 f 是确定性功能，因此我们对确定性功能采用安全性定义，如下所示。

定义 3.令 f：$\{0,1\}^* \times \{0,1\}^*$ $\{0,1\}^* \times \{0,1\}^*$ *为确定性功能。我们说，如果（a）*输出 ^{n} *等于 f（a，b），则 n 在存在静态半诚实对手的情况下安全地计算 f 。（b）存在 PPT 算法* B_1 *和* B_2 *使得（1）* $\{B_1（a，f_1（a，b））\}$ = {视图$^\wedge$（a，b）}，*（2）* $\{B_2（b，f_2（a，b））\}$ = {查看 $_J$（a，b）}，*其中 a*，*b* $\in \{0,1\}^*$ *和*| a | = | b |。

协议。在图 6 中，我们提供了简化的协议，概述了预处理协议的主要结构。

引理 1.没有任何计算上不受限制的对手可以猜出规则 *r* 的概率大于 1 / | R |。输入 R_i。

附录 B.2 提供了此引理的证明。

定理 2.假设 *DDH* 假设成立，则在存在静态半诚实对手的情况下，预处理协议可安全地计算 f 。

该定理的证明在附录 B.3 中提供。

输入：**GW** 具有输入 x，y G Z $_p$；**MB** 具有输入（{R_i， k $_i$}$_{i\,e\,[n]}$），其中 $R_i = g^{w}$，^{i} $^{+}$ ^{ki} 该协议在 **GW** 和 **MB** 之间运行：

1. **GW** 计算 $X = g^{x}$，并将 X 发送 给 **MB**。
2. **MB** 发送{ - [R $_{我}$}$_{我\,e\,[n]}$到 **GW**。
3. **GW** 为 i G [n]计算 X $_i$ =（R $_i$ • g^{y}）$_x$，并将{ X $_i$}$_{ie[n]}$发送给 **MB**。
4. **MB** 计算 K $_i$ = X $_i$ /（X 作为规则 r $_i$的可重用随机规则。

图 6.简化的预处理协议

5.3 令牌加密

它捕获了安全要求，即当给定加密令牌时，**GW** 无法学习基础令牌。

定义。对于消息空间 M，令牌加密方案如下：

- 设置（A）：输入安全参数 A，输出密钥 sk 和公共参数 pk 。
- Enc （pk， sk， t）：将公共参数 pk，密钥 sk 和令牌 t GM 作为输入，输出密文 c 。

安全。它在挑战者 *C* 和对手 A 之间定义。

- **设置**：C 运行**设置**（A），并将公共参数 pk 发送给 A。
- **挑战**：A 从 M 中随机选择两个令牌 t_0， t i 并将它们发送到 C. C 掷出一个随机硬币 b G {0,1}，运行 c Enc （pk， sk， t $_b$），然后将 c 发送给 A。
- **猜测**：A 输出 b 的猜测 b'。

对手的优势定义为 Pr [b ' = b] _1 / 2。

定义 4.*如果没有 PPT 对手在安全游戏中具有不可忽略的优势，则令牌加密方案是安全的。*

建造。下面介绍的结构从安全的角度概述了主要结构。

- **设置**（A）：令 PRF 为伪随机函数。选择随机值 x，y，c，w G Z_p，计算 $p_i = g^{c}$，$= g^{w}$，$p_3 = x$ 和 p $_4$ $= y$。最后，将 c 设置为 sk，将（p $_i$，p $_2$，P 3，p $_4$）设置为 pk 。
- **的** Enc （PK， SK， T）：设**盐** 是随机的盐。计算 t_w = PRF $_{P2}$（t），T_t = g ，c ＝ H（T_t，**盐**）。输出 c 和**盐**。

该定理的证明在附录 B.4 中提供。

5.4 规则隐藏

它捕获了安全要求，即当给定攻击规则元组（由 **RG** 发出）时，**MB** 无法学习基础规则。

定义。对于消息空间 M，规则隐藏方案定义如下：

- **设置**（A）：以安全参数 A 作为输入，输出密钥 sk 和公共参数 pk 。
- RuleHide （PK ， SK ， R）：作为输入公开参数 pk 的，秘密密钥 SK 和一个规则 *r 的 EM ，* 输出一个隐藏规则。

安全。如下所示，在挑战者 *C* 和对手 *A* 之间定义了规则隐藏方案的安全性定义。

- **安装程序**：C 运行**安装程序**（A），并将公共参数提供给 *A*。
- **挑战**：A 从 M 中选择两个随机规则 r_0，r_i，并将它们发送给 C。
  在接收到 r_0和 r_i时，C **掷出**一个随机硬币 *b*，运行 RuleHide （pk ， sk ， r_b）并将返回的隐藏规则返回给 A。
- **猜测**：A 输出 *b* 的猜测 b'。

建造。

- **设置**（A）：令 PRF 为伪随机函数。选择随机 k，$w \in Z_p$，将 g^w设置为 sk ，将 k 设置为 pk 。
- RuleHide （PK ， SK ， R）：计算 $r'_{\text{瓦特}}$= PRF 小号ķ （R），- *[R* = G [R 瓦特 + ķ，并输出 *R.*

定理 4.*假设 PRF 是伪随机函数，即 Sect 中的构造。5.4 是一种安全的规则隐藏方案。*

该定理的证明在附录 B.5 中提供。

6 绩效评估

我们调查客户端和服务器之间网络连接的性能。由于 PrivDPI 的性能优于 BlindBox，因此我们仅介绍与 PrivDPI 的比较。让一个*单轮连接*从客户端到服务器的连接。单轮连接的运行时间反映了客户端可以连接到服务器的速度，而通信成本捕获了为建立此连接而需要传输的开销数据量。理想情况下，单轮连接的运行时间应尽可能短。它花费的运行时间越少，客户端连接到服务器的速度就越快。类似地，期望最小化网络通信开销。我们分别测试了我们的协议和 PrivDPI 的单次连接的运行时间和通讯成本。我们的实验是在 64 位 Linux 操作系统下，运行于 3.20 GHz 且具有 8 GB RAM 的 Intel®Core i7-8700 CPU 上运行的。CPU 支持 AES-NI 指令，其中

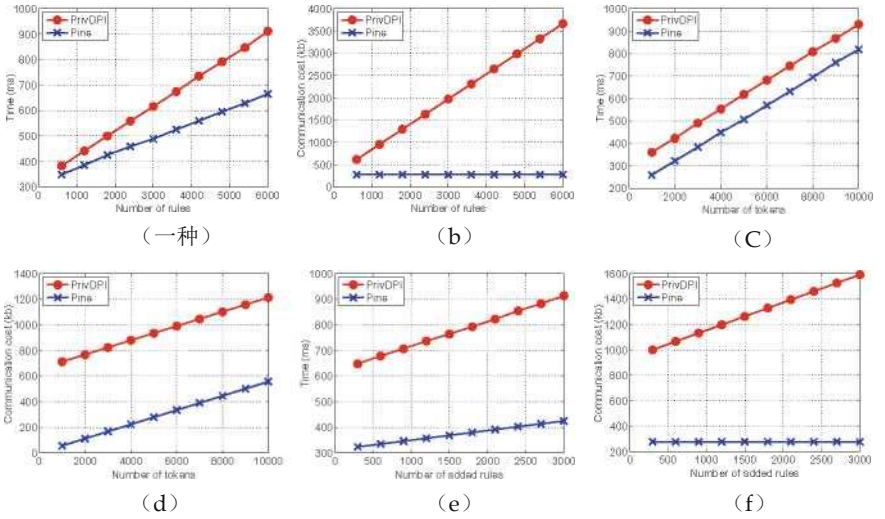

图 7.实验性能

令牌的加密和规则的加密反映了这种硬件支持。实验基于 Charm-crypto [1]，并基于 NIST 曲线 P-256。如第 5 节所述。如图 3 所示，规则和令牌都由 8 个字节组成。为简单起见，我们测试的有效负载不包含重复的令牌。我们对每种情况进行 20 次测试，并取平均值。

规则数量如何影响单轮连接？ 图 7 一 说明了当规则数从 600 到 6,000 时，具有 5,000 个令牌的一轮连接的运行时间。结果表明，与每种情况相比，Pine 花费的时间少于 PrivDPI，规则越多，Pine 花费的时间就越少于 PrivDPI。这意味着 Pine 中的客户端连接服务器所需的时间更少。特别是，对于 5,000 个令牌和 6,000 条规则，Pine 大约需要 665 毫秒，而 PrivDPI 大约需要 912 毫秒。也就是说，Pine 的一轮连接延迟比 PrivDPI 小 27%；对于 5,000 个令牌和 3,000 个规则，Pine 大约需要 488 毫秒，而 PrivDPI 大约需要 616 毫秒。换句话说，Pine 中的客户端连接到服务器的速度比 PrivDPI 快 20.7%。图 7 b 显示规则数为 600 到 6,000 时使用 5,000 个令牌的一轮连接的通信成本。PrivDPI 的通信成本随着规则数量的增长而线性增长，而对于 Pine 而言它是恒定的。规则越多，PrivDPI 产生的通信成本就越高。这是因为 PrivDPI 中的客户端需要运行 **MB** 的预处理协议，并且该预处理协议产生的通信成本与规则数量成线性关系。

令牌数量如何影响单轮连接？ 我们将规则数固定为 3,000，并在令牌数从 1,000 到 10,000 的范围内测试运行时间和通信成本。

图 7 Ç 显示，松树的运行时间是用令牌的有效载荷中的数字，相同 PrivDPI 线性。但是，对于每种情况，Pine 消耗的时间少于 PrivDPI，这是由于以下两个原因。首先，Pine 中的客户端不需要为 3,000 条规则执行预处理协议。第二个原因是，在 PrivDPI 令牌的加密主要发生在一个乘法 *G*，在一个指数 *G*，和一个 AES 加密。在 Pine 中，令牌的加密主要需要执行一次哈希运算，对 *G* 执行一次幂运算和进行一次 AES 加密也就是说，Pine 的令牌加密比 PrivDPI 的令牌加密更快。图 7 d 显示了令牌数量从 1,000 到 10,000 时，具有 3,000 条规则的单轮连接的通信成本。与运行时间类似，Pine 和 PrivDPI 的通信成本都与令牌数量成线性关系，但是 Pine 产生的通信量少于 PrivDPI。这是由于 PrivDPI 中针对 3,000 条规则的预处理协议的额外通信成本。

新添加的规则数量如何影响单轮连接？ 当新添加的规则数量从 300 到 3,000 不等时，我们使用 3,000 个规则和 5,000 个令牌测试了运行时间和通信成本。图 7 Ë 表明，松花费的时间比 PrivDPI 时间更少。对于 3,000 条新添加的规则，Pine 花费 424.96 毫秒，而 PrivDPI 花费 913.52 毫秒。也就是说，Pine 比 PrivDPI 快 53.48％。图 7 ˚F 表明，松的通信成本为小于 PrivDPI。特别是，Pine 的通信成本与新添加的规则数量无关，而 PrivDPI 与新添加的规则数量成线性关系。这是因为 Pine 中的客户端不需要在线执行预处理协议。

7 相关工作

我们的协议是基于 Sherry 等人 提出的 BlindBox 构建的。[20]和 Ning 等人 提出的 PrivDPI 。[15]，如引言中所述。BlindBox 直接引入了对加密流量进行隐私保护的深度数据包检查，而 PrivDPI 利用模糊规则生成机制，与 BlindBox 相比，性能得到了改善。Lan 等人 将 BlindBox 中的构造用作基础组件。[9]进一步提出了 Embark，其利用受信任的企业网关在基于云的中间盒设置中执行隐私保护检测。在 Embark 中，企业网关需要被完全信任并了解流量的内容和检测规则，尽管在这种情况下，客户端不需要执行我们的协议中的任何操作。我们的工作集中在 BlindBox 和 PrivDPI 的原始设置上，通过进一步的性能改进，新属性和更强的隐私保证，同时考虑了实用的企业网关设置，在该设置中，网关无需完全信任。Canard 等。[4]还基于 BlindBox 的概念提出了一种具有更好性能的协议 BlindIDS。该协议包含一个令牌匹配机制，该机制基于基于配对的公钥操作。尽管实用，但它与 TLS 协议不兼容。

另一个相关的工作重点是中间盒的问责制。这意味着客户端和服务器都知道对加密流量执行检查的中间盒，并且能够验证这些中间盒的真实性。Naylor *等*。 [14]首先提出了一种称为 mcTLS 的方案，其中对现有 TLS 协议进行了修改，以实现问责制属性。但是，Bhargavan 等。 [3]表明 mcTLS 可能被攻击者篡改，从而使中间盒所连接的服务器的身份混乱，并且攻击者可能将自己的数据注入网络。因此，提出了一种用于分析此类协议的正式模型。Naylor 等。 [13]进一步提出了一种称为 mbTLS 的方案，该方案不修改 TLS 协议，从而允许对中间盒进行身份验证，而无需替换现有的 TLS 协议。最近，李等*人*。 [10]提出了 maTLS，该协议执行显式身份验证和安全参数验证。

也有一些提案可以分析加密的流量，而无需解密或检查加密的有效负载。机器学习模型用于基于加密流量的元数据检测异常。安德森等。 [2]提出了用于对加密流量进行恶意软件检测的此类技术。可信硬件也已部署用于保护隐私的深度数据包检查大多数建议都利用了英特尔 SGX 的安全区域。主要思想是为驻留在中间盒中的可信硬件提供会话密钥。这些包括 Han 等*人* 提出的 SGX-Box 。[8]，Poddar 等*人的* SafeBricks 。[19]和 Trach 等*人的* ShieldBox 。 [21]和 Duan 等*人的* LightBox。[5]。

我们注意到，我们的工作可以与问责协议以及基于机器学习的工作相结合，以提供涵盖身份验证和隐私的全面加密检查。

8 结论

在本文中，我们提出了 Pine 协议，该协议允许以隐私保护的方式检查加密流量。Pine 在实际设置中建立在 BlindBox 的设置和 PrivDPI 技术的基础上，但与中间的两个工作相比，它可以从中间盒中隐藏规则集，从而显着提高了性能。此外，该协议允许动态添加轻量级规则，就我们所知，以前从未考虑过。Pine 利用通用的实用企业设置，即客户端通过企业网关建立与 Internet 服务器的连接，网关可以帮助建立加密的规则集，而无需了解客户端流量的内容。同时，中间盒检查加密的流量，而无需了解流量的基本规则和内容。通过广泛的实验，我们证明了 Pine 优于 PrivDPI 的性能。我们认为，随着公司和个人用户对隐私的关注日益增加，Pine 是一种检测恶意流量的有前途的方法。

致谢。 这项工作部分得到了新加坡国家研究基金会（NRF2018NCR-NSOE004-0001）和 AXA 研究基金的支持，部分得到了新加坡研究基金会的支持。

国家自然科学基金（61822202，61872089，61902070，61972094，61825203，U1736203，61732021），广东省应用技术研究与关键技术成果转化与发展专项基金（2016B010124009），科技计划（201802010061），部分由国家研究基金会，新加坡总理府根据其企业实验室@大学计划，新加坡国立大学和新加坡电信有限公司提供

中间盒可搜索加密的正确性

一方面，对于每个与规则 r 匹配的令牌，均应以 1 的概率检测到匹配；否则，将检测到匹配。另一方面，对于不匹配 *r* 的令牌，匹配的可能性应该很小。对于每个足够大的安全参数 A 和任何多项式 n（-），使得 n = n（A），对于所有 t i，G *M* ⁿ，对于每个规则 r GM，对于每个满足 r = t，^的索引 ind ⱼ，对于满足 r = t i ₙ d ⱼ的每个索引 ind j，将 Exp ₁（A）和 Exp ₂（A）定义为以下实验：

实验实验 ₁（A）：

sk 设置（A）；（c ₁，...，c ₙ），salt TokenEnc（t ₁，...，*t* ₙ，sk）；e ᵣ RuleEnc （r，sk）；

{ ind ꜰc } ₖE [i] Match （e r，（c ᵦ...，c n），salt）：ind j *e* { ind ꜰc } ꜰce [;]

实验实验 ₂（A）：

sk 设置（A）；（c ₁，...，c ₙ），salt TokenEnc（t ₁，...，*t* ₙ，sk）；e ᵣ RuleEnc （r，sk）；

{ ind ꜰc } ꜰc£ [i] Match （e r，（c ᵦ...，c n），salt）：ind j G {ind ꜰe } ꜰce [i]

我们有 Pr [Exp ₁（A）] = 1，Pr [Exp ₂（A）] = negl （A）。

B 证明

B.1 定理证明 1

通过一种混合证明了安全性，该混合将确定性随机值替换为随机预言。特别是，现在对算法 TokenEnc 进行了如下修改：Hybrid 。TokenEnc （t ₁，...，*t* ₙ，sk）：设 salt 为随机盐。对于 i G [n]的，采样的随机值 ŤĴ在密文空间和集合 Ç，=Ť ᴊ。最后，输出（c ₁，...，c ₙ）和 salt 。算法 RuleEnc （r，sk）被定义为输出随机值 ř 其限制是从密文空间：（1）如果 r 等于吨 ĵ 一些吨 ĵ，-[R 被设定为 ŤĴ；（2）为未来的任何 *R“* 使得 r 等于- *[R* ”，则输出被设定为[R 。我们拥有算法 TokenEnc 和 RuleEnc 的输出是随机的，同时保留了令牌和规则之间的匹配模式。显然，S ₀ = {t ₀，₁，...，*t* ₀，ₙ *}的分布，*小号 ₁ = {吨 ₁，₁，...，T ₁，ₙ }是相同的。因此，任何 PPT 对手都可以更改区分两个正好一半的集合。

B.2 引理 1 的证明

修正一个随机的 $R = g^{r}$，其中 r G Z_{p}。我们认为 $R = R_{i}$的概率是 k_{i} = r r $_{wji}$的概率。因此，对于 VR G G， Pr [R ＝ R $_{i}$] ＝ 1 / p。

B.3 定理证明 2

我们为各方建立模拟器，B $_{Y}$ 代表 **GW**，B $_{2}$代表 **MB**。对于 **GW** 损坏的情况，B_{Y}需要为 **GW** 生成传入消息的视图。**GW** 收到的消息是 R_{i} for i G [n]。为了模拟规则 r $_{i的}$$R_{i}$，B $_{Y}$ 选择一个随机 u_{i} G Z_{p}，计算 $U_{i} = g^{u_{i}}$并将 U_{i}设置为规则 r i 的传入消息，该规则模拟从 **MB** 到 **GW** 的传入消息。继引理 1 之后，**GW** 的模拟传入消息（即 U$_{i}$）的分布与协议的实际执行是无法区分的。接下来，我们考虑 **MB** 损坏的情况。第一条和第三条消息是 **MB** 收到的传入消息。对于第一个消息，B $_{2}$随机选择一个值 v GZ_{p}，计算 $V = g^{v}$，并将 V 设置 为 **MB** 的第一个传入消息。对于第三条消息，B $_{2}$为协议 i G [n]随机选择一个值 v $_{i}$ G Z_{p}，计算 V = $g^{v_{i}}$，并将 V_{i}设置为协议第三步中的传入消息。在协议的实际执行中，规则 r_{i}的 **MB** 视图为（k $_{i}$，R $_{i}$; X，X_{i}）。真实视图和模拟视图的分布为（k_{i}，$g^{f_{随机}r_{随}+k_{随}}$; $克^{x}$，$G^{xR_{随机}，_{随}+xK_{随}+x}v$）和（$K_{我}$，$克^{f_{随机}r_{随}+k_{随}}$;$克^{v}$，$G^{v_{随}}$）。显然，如果 DDH 假设成立，那么 PPT 对手就无法区分这两种分布。

B.4 定理证明 3

我们通过一种混合证明了安全性，在这种混合中，我们用随机值替换了随机预言。特别地，修改后的算法 Enc 描述如下：Enc （pk，sk，t）：设 salt 为随机盐。从密文空间中采样一个随机值 c ∗，并输出 c ∗和 salt 。现在我们有了 Enc 算法的输出是随机的。挑战代币 t_{o}和 t_{i} *的分布*是相同的。因此，没有 PPT 对手有机会区分正好是一半的两个代币。

B.5 定理证明 4

我们通过一个混合动力来证明安全性，该混合动力将 PRF 的输出替换 为随机值。特别是，在挑战阶段，挑战者选择一个随机值 v，计算 R ∗ = g $^{v+k}$（其中 k 是对手公开知道的），然后将 R^{*}返回给对手。显然，如果对手赢得了安全游戏，则可以构建一个模拟器，该模拟器利用对手的能力来破坏 PRF 的伪随机属性。

参考

1. Akinyele，JA 等人：Charm：一种用于快速制作密码系统原型的框架。J.加密工程。**3**（2），111-128（2013）。https://doi.org/10.1007/s13389-013- 0057-3
2. B. Anderson，Paul，S.，McGrew，DA：解密恶意软件对 TLS 的使用（不解密）。J.计算机 毒气。黑客技术。**14**（3），195-211（2018）。https：// doi。 org / 10.1007 / s11416-017-0306-6
3. Bhargavan，K.，Boureanu，I.，Delignat-Lavaud，A.，Fouque，PA，Onete，C . 对 TLS 上负责的代理的正式处理。于：S&P 2018，第 339-356 页。IEEE 计算机学会（2018）
4. Canard，S.，Diop，A.，Kheir，N.，Paindavoine，M.，Sabt，M .：BlindIDS：符合市场标准且对隐私友好的加密流量入侵检测系统。见：AsiaCCS 2017，第 561-574 页。ACM（2017 年）
5. 段洪，王成，袁雪，周艳，王清，任克。：灯箱：以闪电般的速度进行全栈保护的有状态中间箱。于：CCS 2019，第 2351-2367 页（2019）
6. 俄勒冈州 Goldreich：密码学基础：第 2 卷，基本应用。剑桥大学出版社（2009）
7. Google 。 Web 上的 HTTPS 加密 （2019） 。 https： // transparencyreport 。google.com/https/overview?hl=zh-CN
8. Han，J.，Kim，S.，Ha，J.，Han，D .： SGX-Box：使用安全的中间盒模块启用对加密流量的可见性。载于：APNet 2017，第 99-105 页。ACM（2017 年）
9. Lan，C.，Sherry，J.，RA，Popa，Ratnasamy，S.，Liu，Z .：上船：将中间盒安全地外包到云中。于：NSDI 2016，第 255-273 页。USENIX 协会（2016）
10. Lee，H.，et al .: maTLS：如何使 TLS 能够识别中间盒？于：NDSS 2019（2019）
11. Meeker，M .：互联网趋势（2019 年）。https://www.bondcap.com/report/itr19/
12. 国家安全局。管理来自传输层安全检查的风险（2019）。https://www.us-cert.gov/ncas/current-activity/2019/11/19/ 美国国家航空航天局发布网络咨询管理风险运输层安全性
13. Naylor，D.，Li，R.，Gkantsidis，C.，Karagiannis，T.，Steenkiste，P .：还有更多：为两个以上的参与者提供安全的通信。载于：CoNEXT 2017，第 88-100 页 ACM（2017 年）
14. Naylor，D.等人：多上下文 TLS（mcTLS）：在 TLS 中启用安全的网络内功能。在：SIGCOMM 2015，第 199-212 页。ACM（2015 年）
15. Ning，J.，Poh，GS，Loh，JCN，Chia，J.，Chang，EC：PrivDPI：使用可重复使用的模糊规则保护隐私的加密流量检查。于：CCS 2019，第 1657-1670 页（2019）
16. O'Neill，M.，Ruoti，S.，Seamons，KE，Zappala，D。：TLS 检查：多久检查一次，谁在乎？IEEE Internet 计算。**21**（3），22-29（2017）
17. Paxson，V .：兄弟：一种用于实时检测网络入侵者的系统。计算机网络。**31**（23-24），2435-2463（1999）
18. McAfee Network Security Platform（2019）。http://www.mcafee.com/us/products/ 网络安全平台.aspx
19. Poddar，R.，Lan，C.，Popa，RA，Ratnasamy，S。：SafeBricks：屏蔽云中的网络功能。于：NSDI 2018，第 201-216 页。USENIX 协会（2018）
20. Sherry，J.，Lan，C.，RA，Popnas，RA，Ratnasamy，S .： BlindBox：对加密流量进行深度数据包检查。于：SIGCOMM 2015，第 213-226 页（2015）
21. Trach，B.，Krohmer，A.，Gregor，F.，Arnautov，S.，Bhatotia，P.，Fetzer，C .