APQKE-DR：后量子初始化棘轮信道的熵失效分析

Muhammad Asfand Hafeez，研究生会员，*IEEE*，Zain Ul Abideen，会员，*IEEE*，Amna Zafar，Kazi Hassan Shakib，会员，*IEEE*，Arslan Munir，高级会员，*IEEE*

arXiv:2611.32024v1 [cs.CR] 24 Nov 2026

摘要——**面向嵌入式系统和信息物理系统（CPS）的长寿命安全信道必须能够抵御量子攻击和端点随机性故障。尽管已有多种后量子密钥交换协议，但其安全性假设端点具有可靠的随机性，而嵌入式设备并不总能满足这一假设。本文提出熵攻击套件（EAS），用于评估在端点随机性故障（如低熵启动和随机数生成（RNG）路径篡改）情况下的安全信道。作为案例研究，我们设计并实现了 APQKE-DR，这是一种后量子安全信道。其握手使用 ML-KEM [1]和 ML-DSA [2],生成根密钥，供 X25519 双棘轮使用。我们评估了 APQKE-DR 在智能电网通信中的性能和安全性，并将其集成到 ANSI C12.22 和 DNP3 之下，同时不改变有效载荷格式。**

在 Raspberry Pi Compute Module 5 上，APQKE 握手交换 9,610 字节，中位延迟分别为 1.99 毫秒（节点）和 0.82 毫秒（服务器）。棘轮信道每条消息增加 31 字节。在具备新鲜随机性和理想密码原语的情况下，ProVerif 验证了所建模的保密性和身份认证属性。如果随机性失效，EAS 可以从公开协议记录中恢复随机数生成器流和根密钥。在随机数生成器路径篡改下，弱随机数生成器会在多个会话中暴露根密钥。如果篡改还影响 X25519 密钥，攻击者可能获取棘轮密钥和明文。TPM 播种可以防止早期启动攻击，但无法防御后续篡改；而度量启动可以阻止修改后的固件。因此，安全部署要求确保整个熵路径的完整性，而不仅仅是采用强大的密码学原语。

I. 引言

现代嵌入式系统和信息物理系统（CPS）依赖现场设备与集中式控制服务之间的持续通信，传输测量数据至控制软件，并返回驱动系统运行的命令。智能电网将这一模式应用于电力配电：智能电表和其他现场设备向监控与数据采集（SCADA）后端系统发送用量和电网测量数据，后端系统监测状况并协调控制操作 [3]–[5]。这种连接提升了可视性和自动化水平，但也使设备和通信信道暴露于攻击之下，可能导致计费错误、测量数据伪造、可视性丧失或控制功能中断 [6]–[8]。

在这些系统中，每个连接都有两个端点：一个现场设备和一台后端服务器。这些端点会建立会话密钥来保护彼此之间的通信，通常使用 Rivest-Shamir-Adleman (RSA) 或椭圆曲线密码学 (ECC) [9]。能力足够强大的量子对手可以破解这些公钥原语并恢复这些密钥，从而能够解密此前记录的流量密钥 [10], [11]。由于现场设备的能力范围很广，从微控制器单元 (MCU) 级平台到基于 Linux 的网关不等，迁移 a way from these algorithms 很困难；其中资源最受限的设备可用内存容量和计算余量都非常有限。

后量子密钥交换（PQKE）应对量子威胁，但仍要求端点具备不可预测的随机输入 [12]。对于密钥封装机制（KEM），需要随机性来生成临时密钥对，并使用接收方的公钥建立共享秘密。如果端点的随机数生成器（RNG）仅使用过少的不可预测比特进行初始化，或者被修改的软件将其输出替换为可预测序列，则可能违反这一要求 [13]。假设设备对其 KEM 随机输入和用于区分协议运行的握手随机数使用同一个随机数生成器流。如果可能的随机数生成器流集合足够小，可以逐一枚举，攻击者首先为每个随机数生成器流重新生成随机数，并将其与记录的随机数进行比较。对于每个匹配项，攻击者重新生成设备的 KEM 密钥对和封装密文。随后，将重新生成的公钥和密文与记录的握手消息。如果两者匹配，攻击者便使用重新生成的私钥和封装随机性来恢复共享秘密。此搜索需要了解随机数生成器算法以及协议请求随机值的顺序，但无需与设备进行进一步交互 [14]。图 1 展示了智能电表与公用事业后端之间通信中的**量子威胁和端点随机性故障**。

现有协议提供了用于更新加密密钥的经身份认证的密钥建立机制。传输层安全（TLS）1.3 会对其握手转录进行身份认证，并确认派生出的密钥，但其 KeyUpdate 机制会根据前一个流量密钥派生每个新的流量密钥。因而，获取该密钥的攻击者可以在同一连接上派生后续流量密钥 [15]。Signal 协议[16] 通过棘轮消息传递引入新的共享秘密，从而解决了这一限制，其中包括其 Triple Ratchet [17]中的经典更新和后量子更新。然而，其后量子（PQ）扩展 Diffie-Hellman（PQXDH）握手仍依赖经典椭圆曲线密码学来实现双向认证 [18]。

为满足这些身份认证和恢复要求，

M.A. Hafeez、A. Zafar 和 A. Munir 就职于佛罗里达大西洋大学计算机科学与电气工程系，美国佛罗里达州 33431（电子邮箱：mhafeez2024@fau.edu；azafar2025@fau.edu；arslanm@fau.edu）。Zain Ul Abideen 就职于爱达荷大学电气与计算机工程系，美国爱达荷州莫斯科（电子邮箱：zabideen@uidaho.edu）。K. H. Shakib 就职于堪萨斯州立大学计算机科学系，美国堪萨斯州（电子邮箱：kshakib@ksu.edu）。

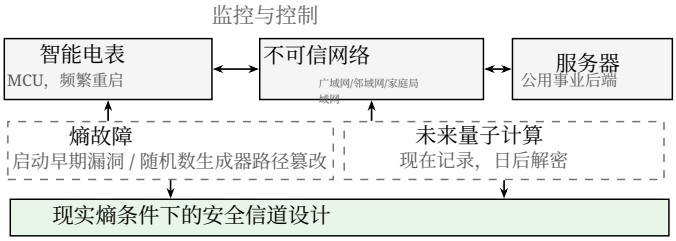

图 1. 智能电表通过不可信网络与 SCADA 通信，面临量子计算机“先记录、后解密”的风险以及实际熵失效问题。

我们设计了 APQKE-DR，将经过身份认证的后量子密钥交换与双棘轮（DR）结合起来，用于节点到服务器的直接通信。我们使用 Raspberry Pi Compute Module 5 作为网关级嵌入式 Linux 端点，在两种受控随机性失效情形下，借助熵攻击套件（*EAS*）评估了该通信信道及其基于硬件的防御：启动早期熵缺失和随机数生成器路径篡改。APQKE-DR 采用后量子方式初始化，但其 X25519 棘轮仅提供经典事后泄露恢复。攻击者失去对信道状态的访问权限后，如果密钥更新引入了攻击者未知的共享秘密，后续消息便可重新获得机密性。我们的贡献如下：

- 我们设计并实现了 **APQKE-DR**，这是一种面向智能电网节点到服务器接口的实验性安全信道。APQKE 握手使用 ML-KEM 进行密钥封装、使用 ML-DSA 生成签名，以实现经过认证的后量子密钥建立 [19], [20],，将派生根密钥绑定到经认证的转录哈希，并采用显式的 *Finished* 风格的 密钥确认[14], [21]。初始根密钥 RK_0 用于初始化 DR 数据通道，并通过密码学方式与经认证的握手转录关联，以供后续消息传递使用。
- 我们引入一种 *EAS* ，用于建模启动早期熵缺失和随机数生成器路径篡改，并展示如何从公开转录中离线恢复根密钥（RK_0），从而能够在熵退化条件下评估握手加信道协议，而非依赖理想化的随机性假设 [22]。
- 我们比较了可信平台模块（TPM）播种与基于平台配置寄存器（PCR）绑定的度量启动门控。在经过测试的启动早期配置中，TPM 播种可阻止恢复，但无法阻止播种后替换随机数生成器输出。在强制执行的启动策略下，PCR 绑定门控会阻止经过测试的修改后固件建立会话。
- 实验评估在网关级测试平台上测量延迟、带宽和系统总功耗，并将 APQKE-DR 集成到 ANSI C12.22 [23] 和 DNP3 [24] 中，同时不改变其应用载荷格式。我们使用理想密码原语和新鲜随机性，在 ProVerif [25] 中针对主动网络攻击者对握手保密性、身份认证和信道绑定进行建模。

本文的组织结构如下：第二节回顾背景与相关工作。第三节介绍 APQKE-DR协议。第四节介绍 EAS 及其实现。第五节涵盖安全性分析和结果。最后，第六节总结全文。

II. 背景、系统与威胁模型

The smar t grid 连接 t资源受限 trained field d

设备通过异构链路进行通信。智能电网中的密钥建立和身份认证协议侧重于最小化端点的计算、带宽和能耗成本，通常使用 ECDH/ECDSA 等椭圆曲线原语，以及对称加密和消息认证码（MAC） [26]。各种轻量级且面向高级计量基础设施（AMI）的设计改进了握手结构，减少状态，并优化消息格式，以适应计量和 SCADA 的约束 [27], [28]。

受“现在记录，日后解密”威胁以及 RSA/ECC 易受量子算法攻击的影响 [10], [11], ，近期工作探索了将后量子原语集成到智能电网和 AMI 安全架构中。原型研究考察了 Kyber 和类似 FrodoKEM 的机制及其在嵌入式AMI部署中的可行性 [29], [30], ，而其他提案则开发了面向智能电网通信的基于格的双向认证和密钥交换协议 [31]。这些研究表明，后量子原语可以被应用于计量基础设施，但其主要关注初始会话建立和系统集成。对于长期生命周期信道上的端到端安全消息传递，包括前向保密、失陷后恢复，以及对现实端点随机性故障的鲁棒性，讨论相对较少。

在智能电网领域之外，安全信道协议为握手记录绑定和显式密钥确认提供了成熟范式，典型例子包括 TLS 1.3（握手记录哈希和 Finished） [14], [21], ，以及 Noise 等模块化握手框架 [32]。对于长期使用的加密消息传递，Signal双棘轮是一种广泛部署的构造，可实现前向保密和失陷后恢复，并已有正式的安全性分析 [33], [34]。另有研究探索了安全消息传递握手的后量子方向 [35]。

A. 系统与威胁模型S

mart-grid nodes $\{SGN_i\}$ 使用加密的 APQKE-DR 通道与公用事业服务器通信 S （图2）。现场设备包括智能电表、远程终端单元（RTU）、相量测量单元（PMU）和网关。AMI 集中器仅转发密文，无法访问通道密钥。现场链路可使用家庭或邻域区域网络，例如家庭局域网（HAN）和邻域网（NAN）。现场链路和回传网络均不可信：攻击者可以观察、篡改或阻止消息。

可信机构为每个节点配置服务器的 ML-DSA验证密钥，并在 S 注册每个节点的验证密钥。握手不会联系该机构。节点可能配备硬件熵源或可信平台模块。软件更新均经过签名，重置后合法的棘轮状态丢失需要进行全新的 APQKE 握手。我们区分六类对手：

- $\mathcal{A}_{\text{网络}}$ 是一个主动的 Dolev-Yao 网络对手，能够观察、修改、注入、重放、重新排序、延迟或丢弃消息；发起并发会话；以及发起计算资源耗尽或状态耗尽型拒绝服务攻击。
- $\mathcal{A}_{\text{种子}}$ 可以重置或断电重启 SGN_i、控制握手时序，并且知道建模的低熵启动种子分布，但无法读取端点机密。

- $\mathcal{A}_{\text{随机数生成器}}$ 只能控制节点的随机数生成器应用程序接口（应用程序接口）所返回的字节。无法访问长期密钥、明文、棘轮状态、可信平台模块状态或其他进程内存。
- $\mathcal{A}_{\text{img}}$ 可以安装并尝试启动修改后的映像，但无法伪造获批准的度量值，也无法绕过强制执行的 TPM PCR 策略。因此，未经授权的镜像会导致节点在信道建立前以故障关闭方式停止运行。
- $\mathcal{A}_{\text{state}}$ 会暂时获取当前的双棘轮状态，包括当前根密钥和链密钥，但之后会失去访问权限，且无法获取配置的密钥或未来的随机性。
- $\mathcal{A}_{\text{Q}}$ 会记录当前的网络流量，并在未来获得一台大型量子计算机，但不会攻陷在线端点。

我们假设配置机构值得信赖，ML-DSA 密钥和密码学实现正确，临时 KEM 秘密及已用消息密钥会被安全擦除，TPM 度量启动策略得到强制执行，且持久凭据和棘轮状态具有完整性。允许合法的状态丢失，但恶意回滚或克隆持久状态不在讨论范围内；任意或持续的运行时内存攻陷、侵入式物理攻击、可信机构或可信平台模块遭到攻陷，以及侧信道攻击或故障攻击也不在讨论范围内。本协议不旨在抵御持续泛洪攻击、彻底的状态耗尽或无限期的消息丢弃。

$\mathcal{A}_{\text{Q}}$ 和 $\mathcal{A}_{\text{state}}$ 针对不同目标。 $\mathcal{A}_{\text{Q}}$ 旨在保护通过 ML-KEM 初始化的记录会话的机密性，使其能够抵御未来的被动量子对手。 $\mathcal{A}_{\text{state}}$ 旨在实现实时棘轮状态遭到泄露后的恢复，但这取决于后续未失陷的 X25519更新，因此仅提供经典事后泄露恢复。

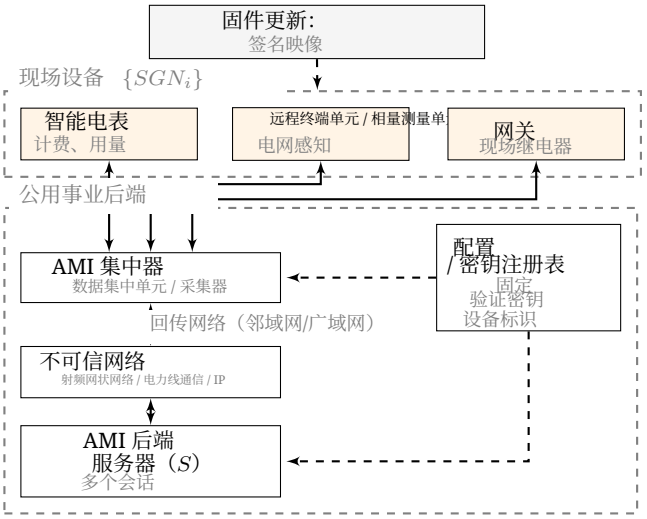

图2。 $\{SGN_i\}$ 通过现场链路和 AMI 集中器连接到后端服务器。配置过程分发信任锚和已注册的验证密钥，而签名更新则修改设备软件。

III. 提议的协议

本节介绍我们提出的 APQKE-DR协议，用于已配置的嵌入式终端 SGN_i 与公用事业后端服务器 S之间的安全通信。该设计包括两部分：**APQKE**，这是一种经认证的后量子握手，可生成初始根密钥 (RK_0)和经认证的转录哈希 (TH_1)；以及**DR** 数据通道，该通道能够感知资源限制，由 RK_0播种，并通过 TH_1 与握手关联，用于应用流量。

为管理资源，我们实现了一种资源感知型重密钥调度。APQKE 在初始连接、状态丢失期间以及此后定期执行。同时，Diffie-Hellman（DH）棘轮刷新机制借助紧凑型消息头持续运行。

A. 可信机构（*TA*）辅助的长期 *ML-DSA* 密钥分发

我们假设存在一个可信机构（TA），在部署前通过带外方式分发长期 ML-DSA 公钥。不同于基于证书的PKI，TA 不会签发证书，也不会在握手期间传输证书材料。相反，TA 将服务器的长期签名验证密钥安全地配置到每台设备中，反之亦然。

具体而言，后端服务器拥有一个长期 ML-DSA 密钥对（sk_S^{sig}， pk_S^{sig}），智能电网的每个节点拥有（$sk_{SGN_i}^{sig}$, $pk_{SGN_i}^{sig}$）。在制造过程中， pk_S^{sig} 被写入每个 SGN_i（密钥固定）中，而服务器则将（ID_i， $pk_{SGN_i}^{sig}$）存储在按设备身份标引的注册表中。这使 APQKE 能够通过使用预先配置的 ML-DSA验证密钥验证签名，来认证所有临时 ML-KEM密钥，而无需在网络上反复传输任何静态公钥。

为实现紧凑的线上身份表示，部署时可使用短密钥标识符，例如 $kid_S = H$ $kid_i = (pk_S^{sig})$ H $(pk_{SGN_i}^{sig})$，以便接收方能够以最小开销选择正确的固定/已注册密钥。

*B. APQKE*握手Fi

P.QKE握手，它建立一个用于 DR 通道的初始 256 位 RK_0 密钥。与基于证书的设计不同，只有长期 ML-DSA验证密钥通过带外方式预先配置，如第三节-A所述；所有 ML-KEM公钥都是临时生成的，并在握手期间交换。

The SGN_i 通过执行步骤 (a)-(c) 发起握手。在步骤 (a) 中，它选择参数 $params$、身份标识ID_i和一个新鲜随机数 N_{SGN_i}，并将其纳入协议记录，以提供新鲜性和抗重放性。在步骤 (b) 中， SGN_i 生成一个一次性 ML-KEM密钥对 ($epm\ sk_{SGN_i}, epm\ pk_{SGN_i}$)，并在完成后将其擦除。在步骤 (c) 中，临时 ML-KEM 公钥使用其长期 ML-DSA签名密钥进行认证。这个提前生成的签名可防止主动攻击者在传输过程中替换 $epm\ pk_{SGN_i}$，因为服务器会在完成消息处理之前检查该签名。签名后， SGN_i 向服务器发送消息 $M_1 = (params, ID_i, N_{SGN_i}, epm\ pk_{SGN_i}, \sigma_{SGN}^{(1)})$。

收到 M_1后，服务器执行步骤 (a)–(i)。在步骤 (a) 中，它使用 ID_i 从其预置注册表中获取 $pk_{SGN_i}^{sig}$ 并验证 $\sigma_{SGN}^{(1)}$；如果验证失败，握手将在建立任何密码状态之前终止。步骤 (b)–(c) 通过采样 N_S 并生成 ($epm\ sk_S, epm\ pk_S$) 来确保服务器的新鲜性。服务器在协议记录中播种

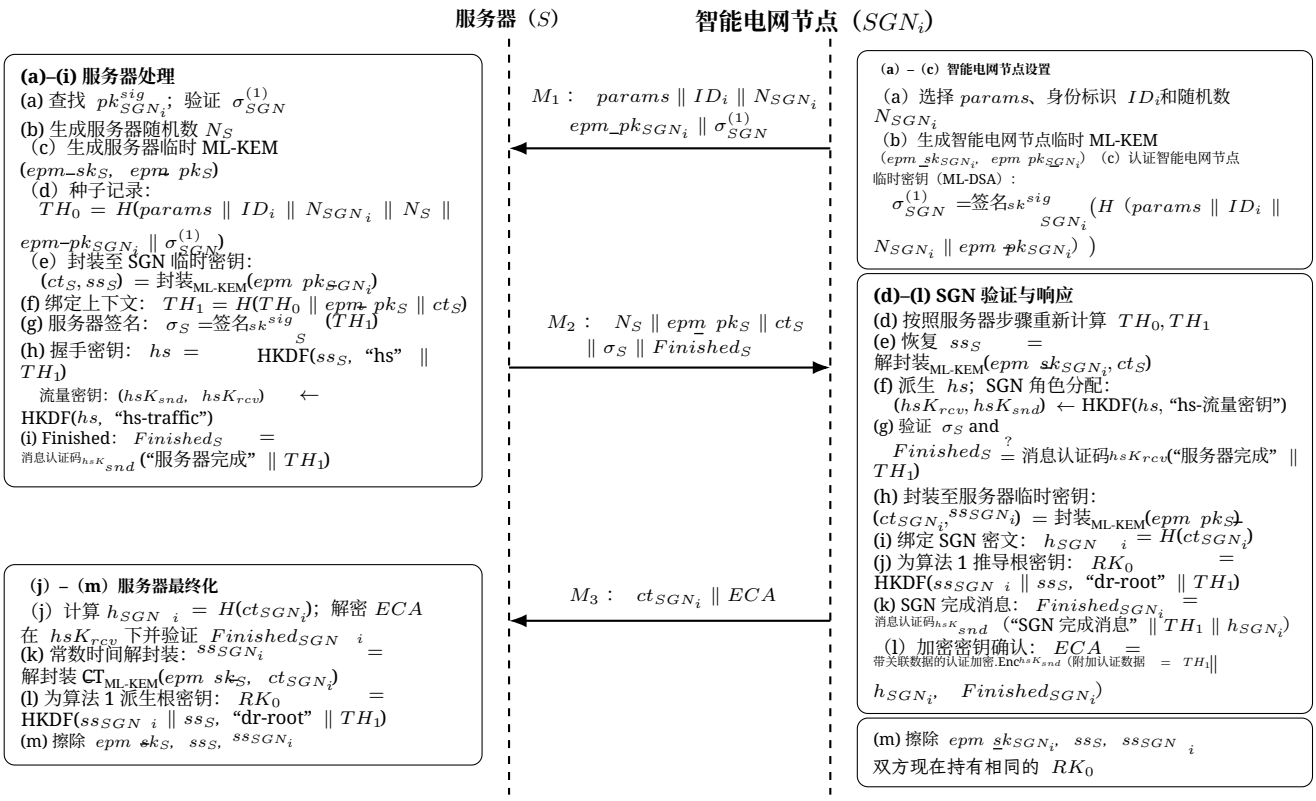

图 3. APQKE（协议 1）：采用临时 ML-KEM 和长期 ML-DSA 进行身份认证，并进行握手记录绑定（TH_0，TH_1）及 Finished 消息认证码，从而为算法 1 生成共享根密钥 RK_0。

步骤 (d) 中将其作为 $TH_0 = H(params \| ID_i \| N_{SGN_i} \| N_S \| epm\_pk_{SGN_i} \| \sigma^{(1)}_{SGN})$，在步骤 (e) 中将其封装至临时密钥，得到 (ct_S, ss_S)，并在步骤 (f) 中以 $TH_1 = H(TH_0 \| epm\_pk_S \| ct_S)$ 完成握手记录绑定。

服务器通过对绑定后的协议记录进行签名来完成身份认证 TH_1（步骤 (g)），并从 ss_S 派生握手流量密钥（步骤 (h)）：$hs = HKDF(ss_S, \text{“hs”} \| TH_1)$ 和 $(hsK_{snd}, hsK_{rcv}) \leftarrow HKDF(hs, \text{“hs-traffic”})$。这是服务器端的分配顺序。$SGN_i$ 派生出相同的两个 HKDF 输出，但以相反顺序分配，即 $(hsK_{rcv}, hsK_{snd}) \leftarrow HKDF(hs, \text{“hs-traffic”})$。因此，$hsK^S_{snd} = hsK^{SGN_i}_{rcv}$ 和 $hsK^S_{rcv} = hsK^{SGN_i}_{snd}$。在此，每个端点都将 hsK_{snd} 视为其发送记录所用的密钥，将 hsK_{rcv} 视为其接收记录所用的密钥。随后，服务器计算 Finished 消息认证码（步骤 (i)），并将 $M_2 = (N_S, epm\_pk_S, ct_S, \sigma_S, Finished_S)$ 发送给 SGN_i。

收到 M_2 后，SGN_i 重构 TH_0，TH_1，将 ct_S 解封装以恢复 ss_S，派生 hs，应用上述 SGN 侧角色分配，并验证服务器签名 σ_S 以及 $Finished_S$？= MAC$_{hsK_{rcv}}$（“服务器完成” $\| TH_1$）。此时，SGN_i 接受服务器，并确认双方就协议记录和流量密钥达成一致。

在完成服务器认证后，SGN_i 通过封装 epm_pk_S 计算第二个 KEM 秘密，并生成（ct_{SGN_i}，ss_{SGN_i}）（步骤（h）），然后使用 $h_{SGN_i} = H(ct_{SGN_i})$ 绑定此密文（步骤（i））。随后，它推导初始双棘轮根密钥（步骤（j））RK_0 = HKDF($ss_{SGN_i} \| ss_S$，“dr-root” $\| TH_1$）。最后，它计算密钥确认 $Finished_{SGN_i}$ = 消息认证码$_{hsK_{snd}}$(“SGN 完成消息” $\| TH_1 \| h_{SGN_i}$)，在步骤（k）中将其加密，并在步骤（l）中将其作为 ECA = AEAD。Enc$^{hsK_{snd}}$(附加认证数据 $= TH_1 \| h_{SGN_i}$, $Finished_{SGN_i}$)，然后将 $M_3 = (ct_{SGN_i}, ECA)$ 发送给 S。

当 S 接收 M_3 时，它计算 $h_{SGN_i} = H(ct_{SGN_i})$，使用 hsK_{rcv} 解密 ECA，并验证 $Finished_{SGN_i}$。随后，它以常数时间解封装 ct_{SGN_i} 以获得 ss_{SGN_i}，派生相同的 RK_0，并擦除临时密钥材料。现在，两个端点都持有相同的 RK_0，同时，经认证的转录哈希 TH_1 可用于将 DR 协议的流量绑定到此次握手。

C. 双棘轮数据通道

DR 通道如算法 1 所述，在 APQKE 握手之后通过跨消息和会话演进密钥，以极低的稳态开销保护智能电网流量。它对引导和 DH棘轮更新都使用 X25519，因此初始化过程具备后量子安全性；然而，为了最大限度地降低开销并减少计算和带宽成本，棘轮信息泄露后的恢复依赖于经典安全假设。该通道以共享的 RK_0 开始，并通过设置 $AD := TH_1$，以及在每次带关联数据的认证加密（AEAD）调用中使用 AD 作为关联数据，与已认证握手建立关联。棘轮信道通过两条引导消息进行初始化，分别记为 B_0和 B_1。

参见算法 1，第 *1* 行 初始化本地 DR状态$\widehat{\ }$，包括 RK_0（根密钥/链密钥、计数器、未设置的对等方DH密钥 pk、空的跳过缓存 $\mathcal{K}_{\text{skip}}$以及 needDH $\leftarrow 0$）。第 *2* 行 从（RK_0、AD）派生短期引导认证器 K_{boot}。在第 *3* 行中，SGN 生成初始 DH 密钥对，并发送 $B_0 =$（pk、τ_C），其中 τ_C 绑定角色标签 “SGN 引导 ”，

算法 1 会话感知双棘轮信道（DH =X25519，带关联数据的认证加密 = ChaCha20–Poly1305）

共享常量： $\kappa = 256$；跳过窗口 W；关联数据 $AD := TH_1$。
状态（每个端点）： RK，CK^{s}，$CK^{\mathsf{r}} \in \{0,1\}^{\kappa}$；计数器 n_{s}，n_{r}，$PN \in \mathbb{N}$；
DH 密钥对（sk，pk）和对等方密钥 $\widehat{pk}$；跳过缓存 $\mathcal{K}_{\text{skip}}$；标志 needDH $\in \{0,1\}$。完整的可变状态记为 $\mathcal{S}$。

引导（协议 1 之后运行一次）：
1. $RK \leftarrow RK_0$；n_{s}，n_{r}，$PN \leftarrow 0$；$\widehat{pk} \leftarrow \perp$；$\mathcal{K}_{\text{skip}} \leftarrow \emptyset$；needDH $\leftarrow 0$。
2. $K_{\text{boot}} \leftarrow \text{HKDF}(RK_0,$ “dr-boot” $\|\ AD,\ \kappa)$。
3. *SGN* $\rightarrow S$：$(sk,\ pk) \leftarrow$ DH。Gen()；$\tau_C \leftarrow$ 消息认证码$_{K_{\text{boot}}}$(“SGN引导” $\|\ AD \|\ pk$)；发送 $B_0 = (pk,\ \tau_C)$。
4. S：验证 $\tau_C \stackrel{?}{=} \text{MAC}_{K_{\text{boot}}}$(“SGN引导” $\|\ AD \|$ ~~pk~~)；设置 $pk \leftarrow pk$；$(sk,\ pk) \leftarrow$ DH。Gen()；$\tau_S \leftarrow \text{MAC}_{K_{\text{boot}}}$（“服务器引导” $\|\ AD \|\ pk$）；发送 $B_1 = (pk,\ \tau_S)$。
5. *SGN*：验证 τ_S ? =消息认证码$_{K_{\text{boot}}}$（“服务器引导” $\|\ AD \|\ pk$）；设置 $\widehat{pk} \leftarrow pk$。
6. 双方计算 $Z_0 \leftarrow \text{DH}(sk,\ \widehat{pk})$，并通过 $(RK,\ CK_0,\ CK_1) \leftarrow \text{HKDF}(RK_0,$ “dr-init” $\|\ Z_0 \|\ AD,\ 3\kappa)$ 派生。
7. 角色分配：SGN 设置 $(CK^{\mathsf{s}}、CK^{\mathsf{r}}) \leftarrow (CK_0、CK_1)$；服务器设置 $(CK^{\mathsf{s}}、CK^{\mathsf{r}}) \leftarrow (CK_1、CK_0)$。擦除 K_{boot}。

头部格式（带宽优化）：
8. 重密钥头：$\text{HDR}_1 = (1、pk、PN、n_{\mathsf{s}})$；数据头部：$\text{HDR}_0 = (0、n_{\mathsf{s}})$。

StartSession()：（由在新会话中首先发送的实体调用）
9. $PN \leftarrow n_{\mathsf{s}}$；$n_{\mathsf{s}} \leftarrow 0$；$(sk,\ pk) \leftarrow$ DH。Gen()；$Z \leftarrow \text{DH}(sk、\widehat{pk})$；$(RK、CK^{\mathsf{s}}) \leftarrow \text{KDF}_{RK}(RK、Z)$；needDH $\leftarrow 1$。

发送(m)：
10. 若 needDH $= 1$，则设置 $hdr \leftarrow \text{HDR}_1$ 并设置 needDH $\leftarrow 0$；否则设置 $hdr \leftarrow \text{HDR}_0$。
11. $(CK^{\mathsf{s}},\ MK) \leftarrow \text{KDF}_{CK}(CK^{\mathsf{s}})$ $nonce \leftarrow$ Encode96(n_{s})。
12. $C \leftarrow$ AEAD.Enc($MK, nonce,$附加认证数据 $= AD \| hdr, m$)；擦除 MK；$n_{\mathsf{s}} \leftarrow n_{\mathsf{s}} + 1$。
13. 发送 MSG $= (hdr, C)$。

接收(hdr, C)：
14. 解析 hdr。如果 $hdr = \text{HDR}_1$，则提取（pk^{peer}，PN^{peer}，n）；否则提取（n）并设置 $pk^{peer} \leftarrow \widehat{pk}$。
15. 设置 $\mathcal{S}' \leftarrow$ Copy($\mathcal{S}$)；第 16–19 行中的操作在 $\mathcal{S}'$ 上执行。如果 $hdr = \text{HDR}_1$ 且 $pk^{peer} \neq \widehat{pk}$，则在 $\mathcal{S}'$ 上运行 **DHRatchet**(pk^{peer}，PN^{peer})。
16. 如果（$\widehat{pk}$，n）$\in \mathcal{K}_{\text{skip}}$，则设置 $MK \leftarrow \mathcal{K}_{\text{skip}}[\widehat{pk}, n]$ 并从临时状态中删除它。
17. 否则，如果 $n > n_{\mathsf{r}} + W$，则丢弃 $\mathcal{S}'$，删除并限制该记录的速率，发出失步信号，并通过经认证的控制路径重新同步。
18. 否则，如果 $n < n_{\mathsf{r}}$，则丢弃 $\mathcal{S}'$ 并将该记录拒绝为重放；否则，当 $n_{\mathsf{r}} \leq n$ 时，执行（CK^{r}，MK）$\leftarrow \text{KDF}_{CK}(CK^{\mathsf{r}})$；
如果 $n_{\mathsf{r}} < n$ 则将 MK 存储为 $\mathcal{K}_{\text{skip}}[pk, n_{\mathsf{r}}]$；$n_{\mathsf{r}} \leftarrow n_{\mathsf{r}} + 1$；结束 while。
19. $nonce \leftarrow$ Encode96(n)；尝试 $m \leftarrow$ AEAD。Dec(MK，$nonce$，AAD $= AD \| hdr$，C)。失败时，丢弃 $\mathcal{S}'$，擦除 MK并中止；成功时，提交 $\mathcal{S} \leftarrow \mathcal{S}'$，擦除 MK并输出 m。

DHRatchet(pk^{peer}, PN^{peer})：
20. 从当前接收链中最多派生并缓存 W 个跳过密钥，直到 PN^{peer}；如果超过此上限，则中止候选棘轮。
21. $\widehat{pk} \leftarrow pk^{peer}$；$n_{\mathsf{r}} \leftarrow 0$。
22. $Z \leftarrow \text{DH}(sk,\ \widehat{pk})$；$(RK,\ CK^{\mathsf{r}}) \leftarrow \text{KDF}_{RK}(RK,\ Z)$。
23. $PN \leftarrow n_{\mathsf{s}}$；$n_{\mathsf{s}} \leftarrow 0$；$(sk,\ pk) \leftarrow$ DH。Gen()；$Z' \leftarrow \text{DH}(sk,\ \widehat{pk})$；$(RK,\ CK^{\mathsf{s}}) \leftarrow \text{KDF}_{RK}(RK,\ Z')$；needDH $\leftarrow 1$。

AD以及在 K_{boot} 下的 SGN 公钥。在第 *4* 行中，服务器验证这一带有角色标签的消息认证码，存储对端 DH 公钥，生成自己的 DH 密钥对，并使用 $B_1 =$（pk、τ_S）进行回复，其中 τ_S 使用不同的标签“服务器引导”。SGN 验证 τ_S（第 *5* 行）后，双方计算初始 DH 共享密钥 Z_0，并派生初始棘轮状态（RK、CK_0、CK_1）（第 *6* 行）。由于 K_{boot} 派生自RK_0，对 B_0 的验证向服务器确认 SGN 已派生出 RK_0；而对 B_1 的验证则明确向 SGN 确认服务器派生出了相同的 RK_0。第 7 行按方向分配链密钥（SGN 使用（CK^{s}、CK^{r}）$\leftarrow$（CK_0、CK_1）；服务器将二者交换），并擦除 K_{boot}。

第 *8* 行 定义了两种头部格式。第一种是用于大多数消息的紧凑数据头部 $\text{HDR}_0 = (0,\ n_{\mathsf{s}})$，另一种是重密钥头 $\text{HDR}_1 = (1,\ pk,\ PN,\ n_{\mathsf{s}})$，其中包含发送方的 DH 公钥和先前发送链的长度 PN，仅用于 DH 更新。

StartSession() 由在新会话中首先发送消息的实体调用。在第 *9* 行，发送方记录其先前发送链的长度（$PN \leftarrow n_{\mathsf{s}}$），重置 n_{s}，生成新的 DH 密钥对，将 DH 共享密钥混入根状态（更新（RK，CK^{s}））并设置 needDH $\leftarrow 1$，以便下一条出站消息携带 HDR_1。

发送应用消息时，第 *10* 行 选择报头：如果 needDH $= 1$，发送方传输 HDR_1 一次并清除 needDH；否则传输 HDR_0。第 *11* 行推进发送链以获取新的一次性消息密钥 MK，并根据当前计数器 n_{s}计算 AEAD 随机数。第 *12* 行 使用关联数据 $AD \| hdr$进行 AEAD 加密，然后擦除 MK 并递增 n_{s}。第 *13* 行传输 MSG $= (hdr,\ C)$。

接收时，第 *14* 行 解析报头以恢复消息索引 n；对于 HDR_1，，还会恢复对端 DH 公钥和对端先前消息编号 PN^{peer}；对于 HDR_0，，接收方则依赖本地存储的对端 DH 公钥 $\widehat{pk}$。第 *15* 行 将完整的接收状态复制到临时候选状态 $\mathcal{S}'$ 中。如果重密钥头包含新的对端 DH 公钥，则 **DHRatchet**过程仅在该候选状态上运行。因此，未经认证的重密钥头无法立即更改持久根密钥、链密钥、计数器、DH 密钥或跳过缓存。

第 *16* 行 检查消息密钥是否存在于候选跳过缓存中，从而允许乱序处理。如果消息索引超出有界跳过窗口，第 *17* 行将丢弃候选状态，丢弃记录并实施速率限制，且仅允许通过经认证的控制路径重新同步；未经认证的报头不再触发协议 1。如果 $n < n_{\mathsf{r}}$ 不存在对应的跳过密钥，第 18 行会将消息拒绝为重放消息。否则，第 18 行会将候选接收链推进至索引 n，并缓存跳过消息所需的中间密钥。随后，第 19 行使用关联数据 $AD \| hdr$执行经认证的解密。

候选状态仅在 AEAD 验证成功后提交；如果验证失败，则会将其丢弃，而不会更改持久化棘轮状态。

如果调用**DHRatchet**，第 *20* 行 会从旧接收链中最多派生并缓存W个剩余密钥，直到 PN^{peer}，若超出该上限则中止候选棘轮。第 *21* 行安装新的对端 DH 公钥并重置接收计数器。第 *22* 行使用当前本地 DH 私钥和新的对端公钥执行 DH 计算，以刷新（RK、CK^{r}）。第 *23* 行 通过生成新的本地 DH 密钥对，立即准备下一个发送纪元，再次刷新（RK、CK^{s}），并设置 needDH $\leftarrow 1$，以便下一条出站消息使用 HDR_1公布新的 DH 公钥。在消息接收期间调用时，这些更新仍为临时状态，只有在关联密文通过 AEAD 验证后才会提交，具体见第 *19* 行。

IV. 拟议熵攻击套件（EAS）

我们在两种 SGN 侧熵故障条件下评估 APQKE 握手，并分析其对 DR 通道的影响：启动早期熵缺失和随机数生成器路径篡改。实验按照第五节所述，在 Raspberry Pi Compute Module 5 上进行。图 4 展示了 SGN 侧的注入点：$\mathcal{A}_{\mathrm{seed}}$ 会影响启动时的确定性随机比特生成器初始化，而 $\mathcal{A}_{\mathrm{rng}}$ 会替换随机数生成器 API 返回的字节；服务器侧随机性保持正常。

在我们的实现中，APQKE 握手所用的所有 SGN 侧随机性（SGN 随机数、ML-KEM 临时密钥生成和封装随机数）均通过图 4 所示的 SGN 平台 RNG 接口获取（由硬件熵源播种的操作系统（OS）密码学安全伪随机数生成器（CSPRNG）/确定性随机比特生成器（DRBG））。

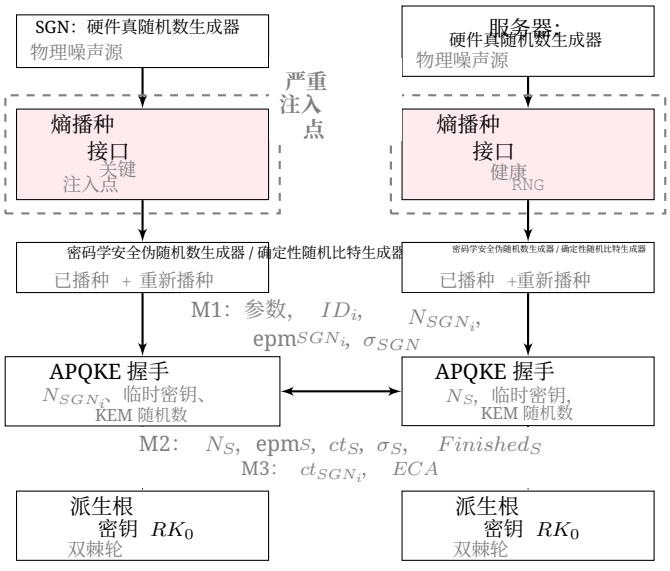

图 4. APQKE 和双棘轮中的熵播种点（消息字段与 APQKE 协议对齐）。

A. 启动早期熵缺失S

设备可能会在网电、维护或更新后重置；如果密码学操作在端点确定性随机比特生成器/密码学安全伪随机数生成器安全初始化之前启动，就会造成早期启动阶段的熵风险 [22]。这种风险是否会出现取决于应用程序随机性接口。在 Linux 上，不带标志的 getrandom() 会等待随机数生成器初始化，而直接读取 /dev/urandom 则可能提前返回数据 [36]。如果攻击者知道或能控制大致的重置时间，那么使用时间戳为应用程序级生成器设定种子，也可能导致其输出可预测 [37]。

APQKE 握手协议要求设备端为 SGN 随机数、临时 ML-KEM 密钥生成以及封装随机数提供随机性。这些输入的可预测性可能暴露握手根密钥 RK_0。为在受控条件下评估这一故障模式，我们在熵边界处实现了启动早期 *RNG* 模式 （即拦截图 4 中的播种接口）。弱 RNG 区间被注入实验平台[1]，并不代表其正常 Linux RNG 经测量存在故障。

该模型由以下参数化：(i) 注入的弱 RNG 区间 T_{inj}，以及 (ii) 大小为2^b 的有效早期启动种子空间（其中 b 为未知早期启动种子位数）。在 $[0, T_{\mathrm{inj}})$ 期间，给定启动种子 $s \in \{0,1\}^b$ 后，RNG 输出流是确定性的；在 T_{inj} 之后，RNG 恢复为健康源。形式化地，令 RNG(·)[2] 输出字节，并令 G 表示设备确定性随机比特生成器 / 伪随机生成器，其中 U ($\{0,1\}^8$) 表示理想化的均匀随机字节。

$$\mathrm{RNG}(t) = \begin{cases} G(s,t), & 0 \le t < T_{\mathrm{inj}}, \\ U(\{0,1\}^8), & t \ge T_{\mathrm{inj}}. \end{cases} \tag{1}$$

启动窗口种子恢复攻击（BWSR）：图 5 展示了攻击者如何利用注入的弱 RNG 区间。攻击者记录一次 APQKE 握手，枚举可能的启动种子，并将重新生成的握手值与协议记录进行比较，以识别出能够重构 RK_0 的种子。攻击者并未破解 ML-KEM 或 ML-DSA；而是利用了早期启动窗口期间较小的有效种子空间。

1) 在线协议记录：攻击者被动记录一次 APQKE 协议运行，该运行在启动后延迟 d 时发起，并观察到（M_1，M_2，M_3）。具体而言，攻击者获知公开的 SGN 随机数 N_{SGN_i} 以及 M_1 中 SGN 的临时 KEM 公钥，随后在 M_2 中观察到（epm pk_S，ct_S），并在 M_3 中观察到 SGN 的密文 ct_{SGN_i}。

2) 离线启动种子搜索：由于（N_{SGN_i}，临时公钥$_{SGN_i}$）源自早期启动随机数生成器流，对手可以针对候选启动种子 $s \in \{0,1\}^b$ 进行离线搜索。对于每个候选种子，对手模拟早期启动随机数生成器的输出，并重新生成相应的值（N'_{SGN_i}，临时公钥$'_{SGN_i}$，临时私钥$'_{SGN_i}$）$\leftarrow$ SimRNG(s), –然后将这些值与 APQKE 握手期间记录的值进行验证。如果重新生成的公开值与观察到的协议记录一致，则接受该种子；具体而言，对 N_{SGN_i} 的前缀匹配可作为高效筛选条件，排除大多数错误候选项。

[1]Raspberry Pi Compute Module 5 作为实验平台 2

我们在启动后经过可配置的延迟 d 启动 APQKE，并测量攻击者是否恢复了 RK_0。记号 RNG(·) 表示设备 RNG 接口。

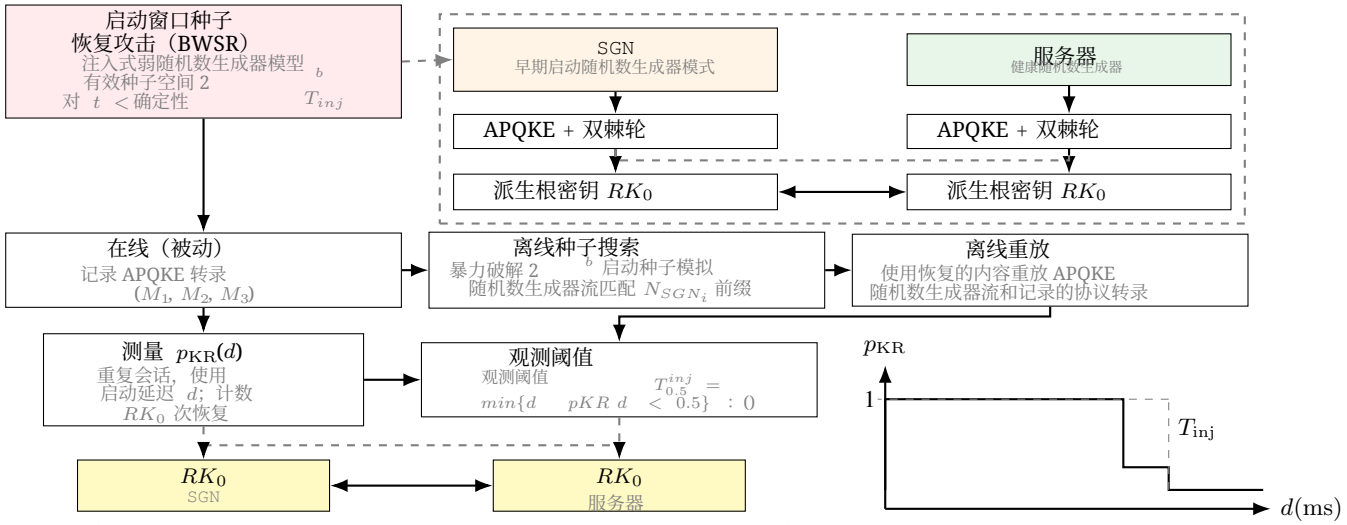

图 5. 启动窗口种子恢复攻击（BWSR）：被动攻击者记录 APQKE 转录，离线搜索 2^b 早期启动种子，并重放握手以恢复 RK_0。

3) 离线 *APQKE* 协议重放与 RK_0 重构：攻击者使用与协议记录一致的任意种子，重现 APQKE 协议中设备端其余的 RNG 抽取（包括 KEM 密钥生成和封装所用的随机币），并重构两个 KEM 共享秘密：(i) 使用 $epm\ sk'_{SGN_i}$ 对记录的 ct_S 进行解封装，获得ss_S；(ii) 针对记录的 $epm\ pk_S$ 重新计算 SGN 封装，获得 ss_{SGN_i}（并通过将由此得到的 ct_{SGN_i} 与记录的密文进行匹配来验证正确性）。由于TH_1 是公开握手值的确定性函数，攻击者可以重构 TH_1，并通过 HKDF($ss_{SGN_i} \parallel ss_S$, "dr-root" $\parallel TH_1$) 推导出相同的会话根密钥 RK_0 =。

这衡量了攻击成功率 $p_{\mathrm{KR}}(d)$，其定义为：在启动后延迟 d 开始的会话中，攻击者恢复 RK_0 的会话所占比例。我们将特征时间定义为$T_{0.5}^{\mathrm{inj}} = \min\{d : \ p_{\mathrm{KR}}(d) \ < 0.5\}$，即攻击者恢复 RK_0 的成功概率降至 0.5 以下时的最短延迟 d。

B. 随机数生成器路径篡改

在 SGN 中，握手需要密码学安全的随机字节来生成其随机数和临时ML-KEM密钥。这些字节来自操作系统随机数生成器 API，最终由硬件熵源播种。我们将从熵源到操作系统生成器再到握手代码的链路称为随机数生成器路径。启动早期熵缺失是该路径起始处的一种瞬态故障：重置后不久，生成器可能使用低熵进行播种，从而使其输出实际上可以被枚举。随机数生成器路径篡改是该路径末端附近的一种持久性故障：受损的固件或软件拦截或替换随机数生成器 API 的输出，将其替换为攻击者可预测的字节，而不论真实熵源如何。随机数生成器篡改是一种已确立的威胁模型，此前已在传输层安全协议实现的背景下进行分析 [38]，并在双棘轮规范 [17]中有所提及。下面将介绍控制随机数生成器 API 输出的对手如何从记录的握手中恢复根密钥 RK_0，以及这种恢复对后续双棘轮消息密钥机密性的影响。第五节-D 报告了实验结果，包括缓解方法。

在所报告的实验中，受限攻击者 $\mathcal{A}_{\mathrm{rng}}$仅控制返回的随机字节。上游熵源、服务器随机数生成器、预置密钥以及其他进程内存均未遭到破坏。受控的 APQKE 提取量包括 SGN 随机数 N_{SGN_i}、临时 ML-KEM 密钥生成随机性和封装随机数。

我们将恶意提供方实例化为一个用于诊断的 16 位线性同余生成器 (LCG)。这个刻意设计得较弱的生成器为实验提供了可复现的失效条件。从 st_0,开始，它会更新并输出

$$\mathsf{st}_{j+1} = (25173\mathsf{st}_j + 13849) \bmod 2^{16}, \tag{2}$$

$$r_j = \mathsf{st}_{j+1} \bmod 2^8. \tag{3}$$

随机数生成器 API 返回字节流 $(r_0, r_1, \ldots)$，以替代正常的随机性。由于 $25173 \bmod 2^8 = 85$ 和 $13849 \bmod 2^8 = 25$，可观测字节满足 $r_{j+1} = (85r_j + 25) \bmod 2^8$.

由于 25 为奇数且 $85 \equiv 1 \pmod 4$，低字节递推具有完整周期 2^8, , 因此其 2^8 个可能的初始值对应同一序列的不同相位。完整的 LCG 有 2^{16} 个内部状态，但共享给定初始低字节的 2^8 个状态会产生相同的协议可见序列。因此，攻击识别的是可观测相位，而非唯一的 16 位状态。攻击者提供了式 (2)– (??) 中的生成器，并且知道其常数以及握手请求随机字节的顺序。利用这些信息，攻击分三步进行（图 6）：记录握手、识别 SGN 使用的相位，并离线重放握手以重新计算 RK_0，无需进一步访问设备。

1) 记录协议记录：攻击者记录 (M_1、M_2、M_3)，并提取 N_{SGN_i}（临时 ML-KEM 公共值）、ct_S 和 ct_{SGN_i}。

2) 流恢复：攻击者将N_{SGN_i} 的第一个字节与 2^8 个可观测流中每个流的第一个随机数单字节进行比较。由于公式 (3) 每个周期会遍历一次所有字节值，因此恰好有一个流匹配。攻击者通过重新生成 SGN 的临时 ML-KEM 密钥对，并将其公钥与 M_1 中的公钥进行比较，确认该匹配。

3) 离线重放：攻击者使用重新生成的私钥解封装 ct_S，以获取 ss_S。随后进行封装

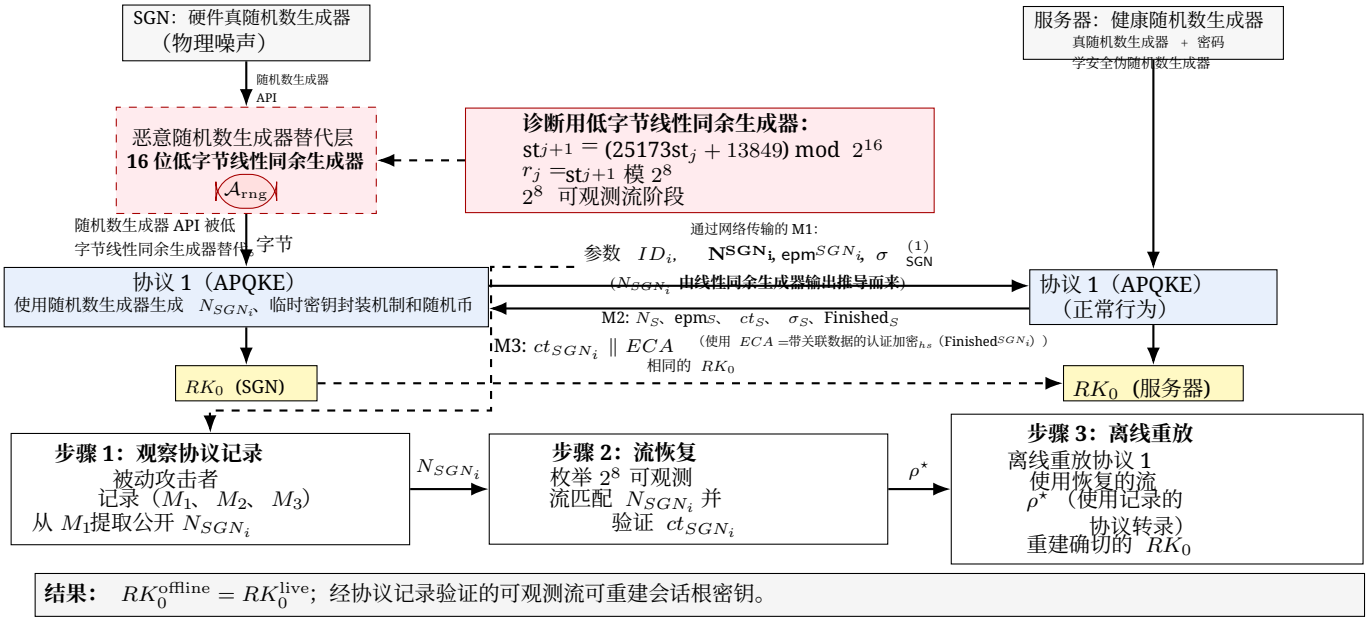

Fig. 6. RNG-路径篡改与离线根密钥 -使用诊断性 16 位低字节线性同余生成器进行密钥恢复

以 epm_pk_S ，从同一流中抽取随机币以获得 ss_{SGN_i}，并检查所得密文是否等于ct_{SGN_i}。它根据 M_1 和 M_2 计算 TH_1 ，并通过 HKDF 派生RK_0 =（$ss_{SGN_i} \parallel ss_S$，“dr-root” $\parallel TH_1$）。这两个共享秘密都可以从 SGN 侧随机数和记录的协议转录中恢复，因此健康的服务器随机数无法保护 RK_0。

恢复 RK_0 后，攻击者可以伪造 K_{boot} 身份验证器，并主动干扰 DR 引导。对于已经完成的诚实引导，这本身不会向被动的经典攻击者泄露链密钥，因为初始化还会混入新的 X25519 秘密值 Z_0，而该值对攻击者仍然未知。如果同一随机数生成器篡改还控制了 X25519 引导或棘轮密钥生成，相应的 DH 秘密将变得可预测，经典攻击者的失陷后恢复仍然无法实现，同时攻击者可以预测后续棘轮更新所引入的秘密。

这些实验评估了随机数生成器路径篡改攻击的缓解措施，并区分种子熵不足与播种后替换随机数生成器输出这两种情况。虽然基于 TPM 的播种会向随机数生成器注入熵，但无法阻止软件替换其输出。在我们的评估中，强制执行的 PCR 策略阻止了使用受评估的修改后固件建立信道。通过针对每种协议的随机数生成器调用序列、握手消息流和密钥派生步骤调整攻击实现，也可以对其他协议进行类似评估。

V. 结果与分析

本节根据第 II-A 节的威胁模型评估 APQKE-DR。我们首先在第 V-A 节给出安全性分析结果，然后分别报告 APQKE 和 DR 的性能测量数据，最后讨论 EAS 的结果及其部署影响。为此，所有实验[3] 均在双节点客户端/服务器测试平台上进行，该平台模拟公用事业主站与现场 SGN 通信。

A. *ProVerif*：安全性分析W

在 ProVerif 中对 APQKE 协议建模 [25]，并采用标准的 Dolev-Yao 攻击者模型：攻击者完全控制网络（拦截、修改、重放和注入消息），而密码学原语则被建模为理想形式（完美的 ML-DSA 签名、完美的 ML-KEM 封装/解封装、抗碰撞哈希，以及 HKDF/MAC/带关联数据的认证加密的标准模型）。APQKE 握手通过使用长期 ML-DSA 验证密钥对临时 ML-KEM 密钥交换进行身份认证，并将整个握手上下文与协议记录的哈希绑定，从而确保通信安全 TH_1。SGN 在确认服务器对 TH_1 的签名并验证服务器消息完整后，接受服务器。服务器仅在验证加密的完成消息并执行解封装后，才接受 SGN。ProVerif 检查的关键目标是派生密钥的保密性 RK_0（以及中间握手密钥的保密性），其中 RK_0 由两个 KEM 共享秘密计算得出，并作为协议记录绑定密钥，由 RK_0 = HKDF($ss_{SGN_i} \parallel ss_S$, "dr-root" $\parallel TH_1$) 派生。ProVerif 证明了保密性查询（攻击者无法推导出 RK_0 或 KEM 共享秘密）以及认证对应性查询（每当某个实体针对 TH_1 触发 接受事件时，必须先发生过一个具有相同身份和协议记录的匹配对端 开始事件）。在符号模型中，这些结果排除了针对握手的经典网络攻击，例如中间人密钥替换、基于重放的协议记录拼接，以及在未破解 ML-DSA/ML-KEM 或伪造 Finished 消息认证码的情况下进行冒充。

DR 数据通道在 ProVerif 中被建模为一个有状态进程，由共享的 RK_0 初始化，并通过设置 RK_0 : 绑定到

[3]服务器配备 32 GB 内存的 Intel Core i9-14900（32 核）上运行 Ubuntu 22.04.5。SGN 端点使用 Raspberry Pi Compute Module 5（Broadcom BCM2712 四核 64 位 Arm Cortex-A76，运行频率为 2.4 吉赫兹），并通过千兆以太网连接。我们通过 SPI 连接了 TPM 9670，将其用作熵失效实验的硬件熵源。

表 I APQKE-DR 与最先进智能电网协议的比较。

属性	APQKE-DR[C]	TLS 1.3[C]	Hybrid TLS[C]	PQXDH [18][H]	PQC-SG [39][H]	Kyber-AMI [29][A]	PUF-Auth [40][A]	LLRA [28][A]	LSEG [27][A]
握手安全									
抗后量子认证密钥建立	✓	×	∼	∼	✓	∼	×	×	×
双向认证和抗重放性	✓	∼	∼	∼	∼	×	✓	✓	✓
握手记录绑定与显式密钥确认 ✓		✓	✓	∼	–	–	–	–	∼
信道安全									
认证的应用数据信道	✓	✓	✓	–	–	✓	∼	∼	✓
逐消息前向保密	✓	∼	∼	–	–	×	×	×	∼
棘轮消息传递	✓	×	×	–	–	×	×	×	×
经典事后泄露恢复	✓	×	×	–	–	×	×	×	×
后量子事后泄露恢复	×	×	×	–	–	×	×	×	×
熵鲁棒性									
已评估端点随机数生成器故障攻击	✓	–	–	–	–	–	–	–	–
已评估熵/启动缓解措施	✓	–	–	–	–	–	–	–	–
实施证据									
智能电网端点/协议集成	✓	–	–	–	∼	✓	∼	∼	✓
后端/网关可扩展性评估	×	–	–	–	✓	∼	–	–	–

✓ = 支持； ∼ = 部分支持或取决于配置； × = 未提供； – = 超出方案声明的范围或未经评估。上标表示主要范围： [C] 完整安全信道、H 握手/会话建立，以及 A 智能电网身份认证与密钥建立。

已认证握手。每个消息密钥都使用对称链式密钥派生函数派生，并明确视为一次性密钥：加密/解密后，消息密钥即被擦除，这符合协议设计——派生的 MK 值在使用后立即删除。每条密文都使用带关联数据的认证加密生成，关联数据包括握手绑定和当前报头（例如，AAD =(TH_1、hdr)），因此任何报头修改，或试图将密文移植到不同的握手转录中，都会被 AEAD 验证检测到。ProVerif 检查：(i) 面对主动攻击者时，应用明文和每条消息的密钥是否保密；(ii) 通过对应断言验证真实性/完整性，这些断言将接收方的 解密接受事件与具有相同报头/计数器上下文的唯一发送方 加密发送 事件关联起来，从而在符号化设定下同时体现完整性和抗重放性。为对前向保密和经典事后泄露恢复进行建模，泄露被表示为在选定时间公开当前棘轮状态。由于过去的 MK 值已被擦除且密钥链持续前进，ProVerif 可以在符号化假设下验证，早先的消息仍然保密。在 ProVerif 模型中，即使攻击者知道当前棘轮状态，只要新的 DH 棘轮步骤引入攻击者仍未知的 X25519 共享密钥，后续消息的机密性就能恢复。因此，这是经典事后泄露恢复的结果，并不涵盖能够破解 X25519 的量子对手。

表 I 比较了握手安全、信道安全、熵鲁棒性和实施证据。APQKE-DR 将 ML-KEM/ML-DSA 握手与握手记录绑定和显式密钥确认、采用逐消息密钥演进的已认证数据通道、基于经典 X25519 的失陷后恢复、包含已测试缓解措施的熵失效评估，以及在 ANSI C12.22 和 DNP3 下的集成结合起来。 [39] 的 PQC-SG 方案提供了强大的后量子智能电网握手，并评估了图形处理

表 II APQKE 握手带宽和延迟（字节）。

Role (已发送)	M1	M2	M3	总计 有效载荷	延迟 第50/95百分位数（毫秒）
SGN	3,690	–	1,144	4,834	1.990 / 3.060
服务器	–	4,776	–	4,776	0.821 / 1.113
总计	3,690	4,776	1,144	9,610	–

器（GPU）网关的可扩展性，但未涵盖长期运行的棘轮信道和熵失效分析。TLS 变体提供完整的传输安全，但不支持持续棘轮演进；PQXDH 则建立初始安全消息传递密钥，并将信道属性留待后续棘轮处理。类似地，LSEG [27] 仅提供经典安全性，缺乏后量子保护。

B. 性能分析

APQKE 开销分析： 表 II 总结了协议 1 的带宽和延迟开销。该协议通过三条消息（M1 至 M3）完成，并输出根密钥 RK_0，用于为算法 1 提供初始密钥。带宽列报告了我们的 APQKE 编码器生成的握手消息的有效载荷大小（不包括传输帧封装）。在此实例中，SGN 发送 M1 和 M3（3,690 B 和 1,144 B），有效载荷总计 4,834 B；服务器发送 M2（4,776 B），因此每次完成的握手有效载荷总计为 9,610 B。消息 M_2 占总有效载荷的 49.7%，包含服务器随机数、ML-KEM 公钥、封装密文、后量子认证以及 Finished MAC。此消息认证码使 SGN 能够验证 SGN 与服务器派生了相同的握手密钥，并且对握手转录达成一致。

延迟以重复握手的第50/95百分位数报告，并使用各端点的时间戳。在 SGN 端，我们将已确认完成时间报告为 $t_{\text{ack}} - t_{\text{start}}$，其中确认是服务器成功验证 M3 并派生 RK_0后发送的基准测试专用确认；这可以避免因异步发送而低估耗时，并提供明确无误的“信道就绪”完成时刻。在服务器端，我们将处理时间报告为 $t_{\text{done}} - t_{\text{m1 接收}}$，涵盖 M1 解析与验证、M2 构造、M3 的接收与处理，_以及最终密钥派生。在我们的测量数据中，SGN 的第50/95百分位数分别为 1.990/3.060 ms，而服务器分别为 0.821/1.113 ms。

DR 带宽开销： 我们在应用程序/传输层边界测量 DR 协议稳态下每条消息的带宽开销。在 DR 实现中，每条应用消息都作为两条成帧记录传输：MSG_HDR（棘轮头），随后是 MSG_DATA（AEAD 密文）。为尽量减少稳态下的带宽占用，MSG_HDR 使用紧凑头部格式，仅携带一个字节的类型标签和一个 32 位消息计数器 n，因此棘轮头有效载荷为 5 字节。较长的重密钥头（包含 pk_{DH}和前一链计数器 PN）仅在会话引导/DH 刷新期间发送，不计入稳态测量数据。

有效载荷使用 ChaCha20-Poly1305 加密，它会将长度为 L 的明文扩展 16 字节，以添加认证标签

表 III DR 通道稳态线上开销与效率随明文载荷大小的变化。

有效载荷字节 (L)	Avg wire bytes/msg (W)	Overhead bytes/msg (E)	Overhead % of Payload (E/L)	Overhead % of total bytes (E/W)	Efficiency % (L/W)
16	47	31	193.8	66.0	34.0
32	63	31	96.9	49.2	50.8
64	95	31	48.4	32.6	67.4
128	159	31	24.2	19.5	80.5
256	287	31	12.1	10.8	89.2
512	543	31	6.1	5.7	94.3

（即，密文长度为 $L+16$）。最后，每条记录都包含一个 5 字节的帧头（type ‖len），因此每条应用消息总共承担 10 字节的传输成帧开销。在所评估的实现中，AEAD 随机数不会在线上传输；它根据每个方向的消息计数器（Encode96(n)）确定性地派生。每条应用消息的线上开销为 $W(L) = L+31$ 字节/消息。线上固定开销为 $E(L) = 31$ 字节/消息，其中包括 5 字节的棘轮头、16 字节的 AEAD扩展，以及 10 字节的传输成帧开销。

表 III 报告了 $W(L)$ 以及随有效载荷大小变化的派生开销/效率指标。我们报告稳态成本，并排除每次运行中仅出现一次的一次性会话引导头（额外的 MSG_HDR 帧，线上大小为 46 字节）；对于长会话而言，这项开销可以忽略不计。

与 ANSI C12.22 和 DNP3 集成： 为展示其在现有智能电网应用协议中的可部署性，我们将 APQKE-DR 作为安全垫片集成在 ANSI C12.22 [23] 和 DNP3 [24]之下。在垫片边界，每个应用协议数据单元 (PDU) 都被视为明文：APQKE 建立 (TH_1, RK_0)，双棘轮从 RK_0 引导，随后 PDU 作为带关联数据的棘轮 AEAD 记录传输，关联数据为 $AD := TH_1$。这使受保护的 SCADA 流量与经认证的握手转录绑定，同时不改变 C12.22/DNP3 格式。

表 IV 总结了 ANSI C12.22 和 DNP3 的开销。在当前原型运行中，服务器生成五个电表读数响应，每个响应包含一个 64 字节的 ANSI C12.19 表载荷（ [23] 中给出的表 27）和一个 154 字节的 DNP3载荷（组30变量1模拟量和组20变量1计数器[24]）。测得的处理开销主要由后量子握手占据：服务器端为 1.926 ms，客户端为 1.872 ms；而 DR 引导的开销为 0.136 ms（服务器）和 0.246 ms（SGN）。保护五个 DNP3 读数（总计 770 B）的成本为发送方加密 0.006 ms，接收方解密 0.004 ms。

最先进 比较： 表 V 展示了 APQKE 握手与具有代表性的基于 PQC 的智能电网协议的客户端执行时间比较。由于仅公开提供了 [41] 和 [39], 的参考实现，因此在与 APQKE 相同的测试平台上执行这两种方案，以获得可直接比较的测量数据。对于

表 IV 采用 APQKE-DR 的 ANSI C12.22/DNP3 集成原型结果（五个读数）。

组件（毫秒）	服务器	SGN
握手	1.926	1.872
DR 引导	0.136	0.246
DR 会话启动	0.102	0.101
DR 数据保护（总计 770 B）	0.006	0.004
总计	2.170	2.223

表 V 使用一阶频率归一化至 2.4 吉赫兹后的客户端计时近似比较。

协议	平台	$f_{\text{rep.}}$ (吉赫兹)	t_{norm} (毫秒)
[44]	树莓派 4 (A72)	1.40	1.17
[42]	树莓派 4 (A72)	1.40	2.04
[43]	树莓派 4 (A72)	1.40	2.39
[39]	树莓派 CM5 (A76)	2.40	2.29
[41]	树莓派 CM5 (A76)	2.40	4.52
APQKE	树莓派 CM5（A76）	2.40	1.99

其余协议，由于其实现不可用或是在不同处理器上测量的，我们按照 CPU 时钟频率进行一阶归一化，单位为 $t_{\text{norm}} = t_{\text{reported}} \times \frac{f_{\text{reported}}}{2.4}$ 吉赫兹。[4]

在 2.4 吉赫兹 Cortex-A76 平台上，APQKE握手的 SGN 中位完成时间达到 1.99 毫秒。其归一化客户端延迟低于以下三种方案： [42], [43] （2.04 和 2.39 毫秒）、 [39] （2.29 毫秒），以及 [41] 的认证配置（4.52 毫秒）。方案 [44] 报告了更低的归一化执行时间，为 1.17 毫秒，但其省略了若干关键安全属性。总体而言，APQKE 的性能处于当代基于 PQC 的智能电网握手的性能范围内。表 V 中的比较仅限于时间指标，因为各方案在参数化方式及其安全功能的完备程度方面存在差异。

C. SGN（TPM 9670）上的硬件锚定熵部

分熵 dpoi点熵 fail措施可以无dermine k估计 bl

嵌入式设备上，我们评估了 SGN 上的一个 TPM 2.0 模块（通过 SPI [45]连接的 TPM 9670），将其作为额外的基于硬件的熵源。该 TPM不 用作批量随机数生成器；相反，仅在会话开始时查询它以获取少量种子数据。图 7 比较了树莓派 SGN 上 TPM2_GetRandom 的输出与操作系统密码学安全伪随机数生成器。32 B 的 TPM 熵请求耗时 0.3 ms，而操作系统随机数生成器读取在 0.1 ms 内完成。作为轻量级合理性检查，我们从每个源收集 8 KB 输出并计算香农熵，得到 7.980 bits/B（TPM）和 7.975 bits/B（操作系统），接近 8.0 bits/B 的最大值。

为量化 TPM 播种是否会影响 APQKE 延迟，我们针对每种设置运行采用 *TPM* 播种熵的 *APQKE* 协议和采用仅操作系统熵的 APQKE 协议，各进行 2000 次已确认握手（N）。我们将 SGN 已确认完成时间报告为（$t_{\text{ack}} - t_{\text{start}}$），其中服务器在成功处理$M_3$ 并派生 RK_0后发送确认。SGN 第50/95百分位数完成时间仍然相近（TPM 播种：2.107/3.188ms；仅操作系统：

[4]这种归一化是近似的，未考虑指令集架构、微架构、存储层次结构或特定于实现的优化方面的差异。

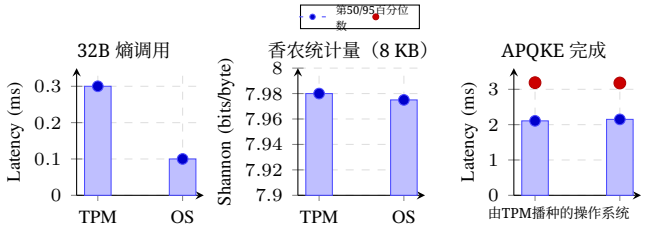

图 7. Raspberry Pi SGN 上的熵源比较（SPI TPM2.0 与操作系统密码学安全伪随机数生成器）：32 字节调用延迟和 8 KB 输出分布合理性检查；APQKE 在 2000 次已确认握手中的第50/95百分位数。

2.149/3.178ms），表明在我们的原型中，注入一个小型硬件锚定种子可以在不造成较大每次握手延迟开销的情况下完成。

*D. EAS*分析

启动早期熵缺失（BWSR）：我们通过在重置后经过受控延迟 d 启动 APQKE，并在第四节-A所述的受控注入模型下的弱随机数生成器模型中进行测试，测量攻击者的密钥恢复成功概率 $p_{\mathrm{KR}}(d)$，从而评估启动早期熵缺失。对于每个延迟设置，我们记录 R 个独立的协议1转录，并运行离线种子恢复过程。随后，我们计算$p_{\mathrm{KR}}(d)$ $=\frac{\text{攻击者恢复密钥的会话}}{R}$ RK_0。为刻画所观察到的转变，我们将 $T_{0.5}$ 定义为经过测试的最早延迟 d，在该延迟下 $p_{\mathrm{KR}}(d)$ 低于固定阈值 $\tau = 0.5$。直观而言，$T_{0.5}$ 将测试区域划分为恢复成功概率高于一半的区域和低于一半的区域。该指标刻画的是受控注入模型；它并不估计底层操作系统密码学安全伪随机数生成器完成初始化所需的时间。

为刻画离线阶段的可行性，我们还报告推导出的种子搜索工作因子[5]：对于有效的早期启动种子空间 2^b 和测得的单次模拟时间t_{try}，预期搜索时间为 $\mathbb{E}[T_{\mathrm{search}}] \approx 2^{b-1} t_{\mathrm{try}}$。

在 SGN 上，我们在固定延迟 d 后启动 APQKE，并将弱 RNG 区间设为 T_{inj}。对于每个 d，我们收集$R = 20$ 次独立握手，并对每份协议记录运行离线攻击者。我们评估：(i) 仅操作系统，在 SGN 随机数生成器接口注入早期启动随机数生成器故障；以及 (ii) *TPM* 播种，在 APQKE 运行前将硬件派生种子材料混入会话级随机数生成器状态。

图 8(a) 绘制了 $p_{\mathrm{KR}}(d)$ 随握手启动延迟的变化。在仅操作系统配置下，在弱 RNG 区间内启动的握手仍可可靠地恢复密钥；当延迟约为 $d = 2000$ ms 时，恢复率降至 $\tau = 0.5$ 以下。此时，在 $R = 20$ 次试验中仅有少数几次成功恢复，而延迟更长时则未观察到恢复。在此阈值以下，注入实验的结果为 $T_{0.5}^{\mathrm{inj}} \approx 2000$ ms。在 TPM 播种配置下，攻击者始终无法恢复根密钥（对于所有 d，$p_{\mathrm{KR}}(d) = 0/20$），因此为 $T_{0.5}^{\mathrm{tpm}} = 0$ ms。

这项受控实验表明，弱早期启动随机数生成器状态（例如由重复重置或不完整的熵初始化导致）可能使记录的 APQKE 转录可用于离线重建 RK_0。$\approx$ 2000 毫秒的转变仅适用于我们注入故障的设置，并非对 Raspberry Pi OS 随机数生成器漏洞或初始化时间的测量。在此模型中，TPM 播种消除了可预测的启动时间依赖性。

结果：随机数生成器路径篡改（线性同余生成器）：我们通过在 SGN 侧的随机数生成器 API 插入恶意垫片（即替换随机数生成器 API 输出的插入层），并强制 APQKE 协议消耗的所有设备端随机性均由确定性 16 位线性同余生成器生成，评估了随机数生成器路径篡改。我们针对每种 SGN 配置执行了 $R = 2000$ 次 APQKE 握手，并记录公开协议记录（特别是 M_1，其中包含 N_{SGN_i}）以及生成的根密钥材料。我们在同一平台上比较了三种构建：(i) 仅操作系统 随机性（USE_可信平台模块=0），(ii) *TPM* 播种随机性（USE_可信平台模块=1），以及 (iii) *TPM-PCR*（图 8(c) 中的 TPM-PCR），它会在 APQKE 使用任何随机性之前添加一个失败即关闭 的 TPM 门控。具体而言，SGN 会向 TPM 请求一个短密钥 K_{gate}（16 字节）；该值会密封 到平台的度量启动状态（TPM-PCR 寄存器用于存储启动组件的哈希）。如果启动状态与已知良好基线匹配，TPM 就会释放 K_{gate}；否则，请求将失败。随后，SGN 会将 K_{gate} 异或混入 APQKE 使用的握手级种子中（用于随机数生成、ML-KEM 密钥生成和封装随机数），并在无法获取 K_{gate} 时中止握手。在 (i) 和 (ii) 中，线性同余生成器垫片位于播种源（操作系统或 TPM）之后，因此无论种子最初来自何处，它都会替换返回给 APQKE 的字节。

垫片维护一个名义上的 16 位内部寄存器，如图 8(b) 所示，并按 $\mathrm{st}_{j+1} = (a \cdot \mathrm{st}_j + c) \bmod 2^{16}$,更新，参数为 $(a,\ c) = (25173,\ 13849)$，每次输出更新后状态的低 8 位作为字节。由于协议只能观察到字节输出，低字节投影仅产生 2^8 个不同的可观测流。观察到 N_{SGN_i} 的第一个字节后，攻击者便可直接识别相应的可观测流。尽管仍有 2^8 个内部状态与该字节一致，但它们构成一个等价类，其中的元素会生成完全相同的输出流，因此不会提供额外的攻击防护。图 8(b) 展示了这种内部状态别名现象。因此，这个线性同余生成器被用作一个刻意设计的弱 8 位可观测诊断模型，用于验证端到端的密钥恢复传播，并评估缓解机制的有效性及其部署位置。

给定一份记录的协议转录，攻击者会识别出与观测到的 N_{SGN_i}前缀一致的可观测流代表，并在离线状态下重放 APQKE 协议，同时强制使用相同的确定性随机数生成器流。这样便可重新生成 SGN 中依赖随机数生成器的计算，包括随机数生成、临时 ML-KEM 密钥生成和封装随机性，从而重建 ML-KEM 共享秘密，进而恢复 APQKE 派生的根密钥。攻击者无需恢复唯一的 16 位内部状态，因为已识别等价类中的每个状态都会生成相同的、协议可见的字节流。

在记录的 $R = 2000$ 次运行中，攻击者以压倒性的概率得手，如图 8(c) 所示；这对仅操作系统构建和 TPM 播种构建均成立。重要的是，在此威胁模型下，TPM 播种并不会降低 p_{KR}，因为子版本发生在 TPM/操作系统熵源之后；一旦随机数生成器

[5]攻击者会从大小为 2^b 的空间中测试可能的早期启动种子 s。每次尝试需要 t_{try} 秒。工作因子是找到一致种子所需的预期试验次数或总时间。

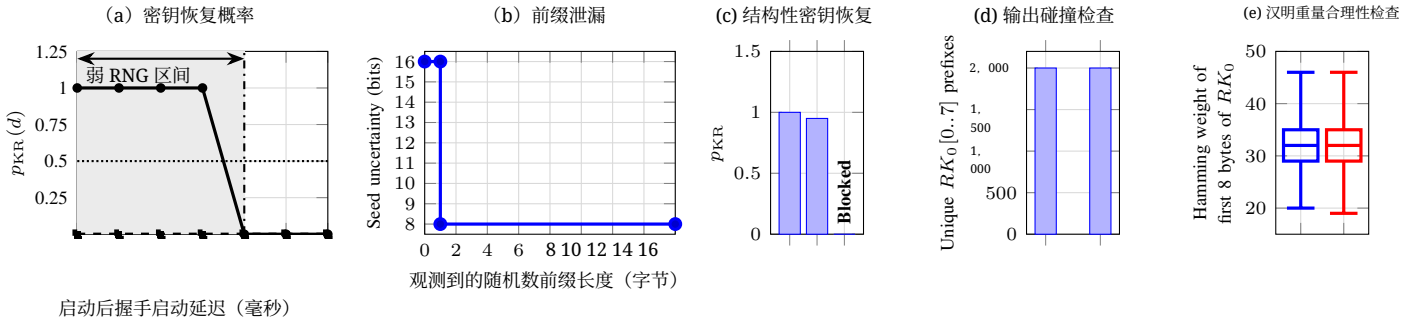

图 8. (a) $p_{KR}(d)$ 与握手启动延迟的关系，(b) 低字节 LCG 混叠：一个随机数单字节将 2^{16} 个状态缩减为一个含 2^8 个状态的等价类，(c) 随机数生成器路径篡改下的结构性密钥恢复，(d) 输出碰撞检查：两种配置中观测到的所有 $R = 2000$ 个 $RK_0[0..7]$ 前缀均不相同，(e) 对 $R = 2000$ 次握手中的 $RK_0[0..7]$ 进行汉明重量合理性检查。

应用程序接口被替换，协议便会消耗攻击者控制的字节，而不受上游熵质量影响。相较之下，在 TPM PCR 门控机制下，LCG 垫片固件会改变 TPM PCR 中反映的启动度量值，因此 TPM 拒绝释放 K_{gate}。如果 K_{gate} 不可用，TPM PCR 构建会采取故障关闭策略，SGN 会在完成 APQKE 之前中止，并且在该垫片镜像下不会建立任何已接受的会话。

一种天真的仅输出检查方法是统计各次握手中重复出现的 RK_0 值。在我们的日志中（每种配置 $R = 2000$ 条），仅操作系统运行和 TPM 播种运行中观测到的所有 $RK_0[0..7]$ 前缀均不相同，如图 8(d) 所示。服务器生成的值和公开协议记录会随会话变化，因此，即使 SGN 侧的随机数生成器被限制为仅有 2^8 条可观测流，HKDF 仍可能生成彼此不同且表面上分布良好的根密钥。然而，对于任何一份记录的协议转录，攻击者仍只需枚举至多这些由字节组成的 2^8 条可能流，并使用公开随机数和 KEM 密文验证正确的一条。因此，没有重复的 RK_0 前缀并不意味着能够抵御协议转录引导的根密钥恢复。[6]

为说明分布合理性检查可能产生误导，图 8(e) 和图 9 显示，$RK_0[0..7]$ 的汉明重量在仅操作系统运行和 TPM 播种运行之间看起来在统计上相似。尽管表面上看似正常，确定性重放仍可离线重建 RK_0，因为弱点是结构性的，而非 RK_0 输出中可检测到的偏差。不过，TPM-PCR 缓解措施并不依赖输出的“合理性”；在垫片镜像下，它会以故障安全方式拒绝建立任何会话。

恢复 RK_0 后，便可伪造 K_{boot} 验证器，进而主动干扰 DR 引导。单凭这一点并不能泄露已完成的诚实引导所产生的链密钥，因为初始化过程还会混入新鲜的 X25519 秘密 Z_0。如果随机数生成器篡改影响了 X25519 密钥生成，DH 秘密就会变得可预测，失陷后恢复也会受到破坏。

E. 讨论与局限性

APQKE-DR 为智能电网端点提供了一种实用的后量子初始化安全信道。ProVerif 分析表明，APQKE 握手能够保护 RK_0 机密性，并抵御密钥替换和重放等主动攻击。由于 APQKE 仅在设置阶段运行（例如初始关联时或状态丢失后），其通信开销会分摊到长期运行过程中。在

[6]TPM-PCR 会在密钥建立前阻止修改后的映像；因此，受攻击会话不会产生 RK_0 样本，相应指标报告为已阻止。

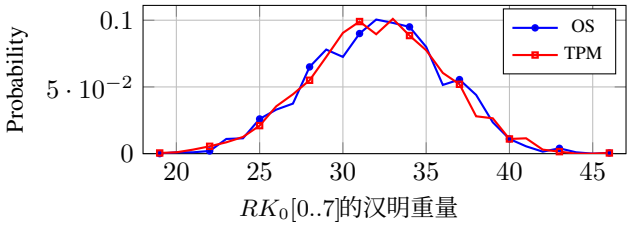

图 9。OS 和 TPM 在 $R = 2000$ 次握手中 $RK_0[0..7]$ 的汉明重量的经验概率质量函数。

稳态下，双棘轮每条消息增加 31 B 的开销，仅对非常小的 PDU 影响显著，随着有效载荷增大，这种开销可以忽略不计。ANSI C12.22 和 DNP3 垫片测试表明，后量子握手是设置成本的主要部分，而棘轮保护仍然轻量，无需更改应用层。

尽管我们在智能电网环境中实现并评估了 APQKE-DR，但其核心构造并不局限于 ANSI C12.22、DNP3 或任何特定的智能电网架构。它适用于具有预配置的端点身份、受管理的后端、长期存在的加密会话、可能发生的重启或状态丢失，以及需要将后量子初始密钥建立与轻量级逐消息密钥演进相结合的部署场景。示例包括工业控制和 SCADA、水气计量、远程环境监测、边缘物联网和医疗设备、交通基础设施，以及其他关键基础设施。在这些场景中，APQKE-DR 可以像我们的智能电网原型一样，通过安全层保护现有的应用载荷。不过，在大规模实施之前，每个领域都需要进行特定领域验证。

EAS 结果表明，长会话安全取决于端点随机性的质量和随机数生成器路径完整性。在受控早期启动模型中，注入的弱随机数生成器窗口期间可以恢复 RK_0，其时间为 $T_{0.5}^{\text{inj}} \approx 2000$ 毫秒，位于$\tau = 0.5$。TPM 播种可在测试的各个延迟下阻止恢复，且对延迟的影响极小，但无法阻止下游随机数生成器替换：在这种情况下，APQKE 会接受攻击者控制的字节，而仅输出检查仍可能看起来正常。

六、结论

我们介绍了 APQKE-DR，并使用所提出的 EAS 评估了其性能以及对端点随机性故障的韧性。该协议将经过身份认证的后量子密钥建立与 X25519 双棘轮相结合，其失陷后恢复依赖于经典假设。该原型集成于 ANSI C12.22 和 DNP3 之下，无需更改应用载荷格式。其实现了

中位确认握手完成时间为 1.990 ms，并为每条消息增加 31 B。在新鲜随机性和理想密码原语条件下，ProVerif 验证了模型中的保密性和身份认证属性。

EAS 实验表明，当端点随机性失效时，这些属性可能会受到破坏。在测试的情形中，可预测的随机性使得攻击者能够从公开握手记录中恢复初始根密钥。TPM 播种在测试的早期启动情形中阻止了恢复，但未能阻止播种后进行随机数生成器替换。度量启动门控阻止了测试中的修改后固件建立会话。

参考文献

[1] G. Alagic、Q. Dang、D. Moody 等，“模格基密钥封装机制标准”，2024 年。[2] T. Dang、J. Lichtinger、Y.-K. Liu 等，“模格基数字签名标准”，2024 年。[3] J. Gallegos、P. Ar´evalo、C. Montaleza 等，“可持续电气化——配电网络的进展与挑战：综述”，《可持续性》，第 16 卷，第 2 期，第 698 页，2024 年。[4] M. Amin，“能源基础设施在可靠性、安全性、效率和韧性方面的挑战：迈向智能自愈电网”，载于《2008 IEEE 电力与能源学会大会——21 世纪电能的转换与输送》。IEEE，2008 年，第 1–5 页。[5] E. Kabalci 和 Y. Kabalci，“智能电网架构简介”，载于《智能电网及其通信系统》。施普普林格，2018 年，第 3–45 页。[6] M. Z. Gunduz 和 R. Das，“智能电网上的网络安全：威胁与潜在解决方案”，《计算机网络》，第 169 卷，第 107094 页，2020 年。[7] S. Goel 和 Y. Hong，“智能电网实施中的安全挑战”，载于《智能电网安全》。施普林格，2015 年，第 1–39 页。[8] 爱达荷国家实验室，“美国电力行业的网络威胁与漏洞分析”，2016 年 8 月。 [在线]。可访问：https://www.en ergy.gov/sites/prod/files/2017/01/f34/Cyber%20Threat%20and%20Vuln erability%20Analysis%20of%20the%20U.S.%20Electric%20Sector.pdf [9] Y. Yan，“椭圆曲线密码学（ECC）概述”，IOP Publishing，第 012019 页，2022 年。[10] H. Y. Wong，“Shor 算法”，载于《量子计算导论：从外行到程序员的 30 个步骤》。施普普林格，2023 年，第 289–298 页。[11] A. A. Giron，“将应用迁移到后量子密码学：超越算法替换”，Cryptology ePrint Archive，2023 年。[12] V. Piscopo，“面向后量子密码学的真随机数生成器设计”，博士学位论文，都灵理工大学，2024 年。[13] D. Han[1], Y. Ma、T. Chen 等，“物理层密钥生成的最小熵估计：一项实证研究”，载于《信息安全与密码学：第 21 届国际会议，Inscrypt 2025，2025 年 10 月 19–22 日，修订精选论文，第三部分》。施普林格·自然，2026 年，第 404 页。[14] E. Rescorla，“RFC 8446：传输层安全协议（TLS）1.3 版”，IETF，2018 年。 [在线]。可访问：https://www.rfc-editor.org/rfc/rfc8446 [15] E. Rescorla，“传输层安全协议（TLS）1.3 版”，技术报告，2018 年。[16] K. Cohn-Gordon、C. Cremers、B. Dowling 等，“Signal 消息协议的形式化安全分析”，《密码学杂志》，第 33 卷，第 4 期，第 1914–1983 页，2020 年。[17] T. Perrin 和 M. Marlinspike，“双棘轮算法”，GitHub wiki，第 112 卷，第 4 期，2016 年。[18] E. Kret 和 R. Schmidt，“pqxdh 密钥协商协议”，Signal，2023 年。[19] 美国国家标准与技术研究院，“FIPS 203：模格基密钥封装机制标准（ML-KEM）”，《联邦信息处理标准出版物》，2024 年。 [在线]。可访问：https://csrc.nist.gov/pubs/fips/203/final [20] 美国国家标准与技术研究院，“FIPS 204：模格基数字签名标准（ML-DSA）”，《联邦信息处理标准出版物》，2024 年。 [在线]。可访问：https: //csrc.nist.gov/pubs/fips/204/final [21] B. Dowling、M. Fischlin、F. G¨unther 等，“TLS 1.3 握手协议候选方案的密码学分析”，载于《第 22 届 ACM SIGSAC 计算机与通信安全会议论文集》，2015 年，第 1197–1210 页。

[22] 美国国家标准与技术研究院，“随机比特生成所用熵源的建议”，NIST 特别出版物 800-90B，2018 年。 [在线]。可访问：https: //csrc.nist.gov/publications/detail/sp/800-90b/final [23] A. Shees、M. A. Taher 和 A. Sarwat，“高级计量基础设施中用于安全智能电表通信的 ANSI C12.22 协议性能分析”，《IEEE 工业电子学会开放期刊》，第 6 卷，第 1831–1847 页，2025 年。[24] G. Clarke、D. Reynders 和 E. Wright，《实用现代 SCADA 协议：DNP3、60870.5 及相关系统》。Newnes，2004 年。[25] B. Blanchet、B. Smyth、V. Cheval 等，“ProVerif 2.00：自动密码协议验证器、用户手册与教程”，版本说明，第 16 卷，第 05–16 页，2018 年。[26] L. Nkenyereye、A. Thakare、P. Khataniar 等，“智能电网轻量级认证协议：面向资源受限型智能电表的节能认证方案”，Mathematics，第 13 卷，第 580 页，2025 年 4 月。[27] A. Zafar、M. A. Hafeez 和 A. Munir，“LSEG：面向智能电网通信的轻量级安全密钥交换协议”，arXiv 预印本 arXiv:2511.07548，2025 年。[28] R. Cheng、Y. Yang、Z. Zhang 等，“LLRA：面向智能电表的轻量级抗泄漏认证密钥交换方案”，《IEEE 智能电网汇刊》，2024 年。[29] V. Lagrota、M. d. L. Filomeno、L. de MBA Dib 等，“一种抗量子的高级计量基础设施”，载于《巴西信息安全与计算机系统研讨会（SBSeg）》。SBC，2025 年，第 32–48 页。[30] V. L. Da Costa、J. L´opez 和 M. V. Ribeiro，“面向智能电表的后量子密码方案片上系统实现”，载于《第 39 届巴西电信与信号处理研讨会——SBrT》，2021 年，第 26–29 页。[31] H. Shekhawat 和 D. S. Gupta，“面向智能电网的抗量子格基双向认证与密钥交换方案”，《新兴电信技术汇刊》，第 35 卷，第 5017 页，2024 年 7 月。[32] T. Perrin，“Noise 协议框架”，规范，2018 年。 [在线]。可访问：https://noiseprotocol.org/noise.html [33] J. Alwen、S. Coretti 和 Y. Dodis，“双棘轮：Signal 协议的安全概念、证明与模块化”，载于《密码学理论与应用年度国际会议》。施普林格，2019 年，第 129–158 页。[34] C. Cremers、C. Jacomme 和 A. Naska，“安全消息传递中的会话处理形式化分析：将安全性从会话提升至会话串”，载于《第 32 届 USENIX 安全研讨会（USENIX Security 23）》。USENIX 协会，2023 年，第 1235–1252 页。[35] J. Brendel、K. Fiedler、F. G¨unther 等，“后量子异步可否认密钥交换与 Signal 握手”，《密码学杂志》，第 35 卷，第 3 期，第 17 页，2022 年。[36] S. M¨uller，“熵源与确定性随机数生成器管理器”，2024 年。[37] D. Eastlake 三世、J. Schiller 和 S. Crocker，“安全性的随机性要求”，技术报告，2005 年。[38] S. Checkoway、R. Niederhagen、A. Everspaugh 等，“TLS {实现中双} EC {的实际可利用性”，载于《第 23 届 USENIX 安全研讨会（USENIX Security 14）》，2014 年，第 319–335 页。[39] M. A. Hafeez 和 A. Munir，“一种面向智能电网的基于后量子密码学的认证与密钥建立协议”，《ACM 信息物理系统汇刊》，2026 年。[40] H. M. S. Badar、S. Qadri、S. Shamshad 等，“使用物理不可克隆函数的智能电网环境身份认证协议”，《IEEE 智能电网汇刊》，第 12 卷，第 5 期，第 4426–4434 页，2021 年。[41] K. H. Shakib、M. A. Hafeez 和 A. Munir，“AmphiKey：面向智能电网的双模式安全认证密钥封装协议”，arXiv 预印本 arXiv:2509.01701，2025 年。[42] H. Shekhawat 和 D. S. Gupta，“面向智能电网的抗量子格基双向认证与密钥交换方案”，《新兴电信技术汇刊》，第 35 卷，第 5017 页，2024 年 7 月。[43] H. Shekhawat 和 D. S. Gupta，“面向智能电网计量基础设施的抗量子区块链辅助无证书数据认证与密钥交换方案”，《普适与移动计算》，第 100 卷，第 101919 页，2024 年。[44] B. Bera 和 B. Sikdar，“保障智能电网应用的后量子通信安全”，载于《2024 IEEE 智能电网通信、控制与计算技术国际会议（SmartGridComm）》。IEEE，2024 年，第 555–561 页。[45] M. Festa，“将自主主权身份锚定至硬件 TPM 2.0”，博士学位论文

，都灵理工大学 ，2025年。