arXiv:2603.02690v1 [cs.CR] 3 Mar 2026

VA-DAR：一种支持后量子密码学、厂商无关的确定性工件解析，用于无服务器、抗枚举钱包恢复

Jian Sheng
Wang Yeah LLC
jason@yeah.app

2026年3月3日

摘要

无服务器钱包恢复必须在可移植性、可用性与隐私之间取得平衡。公开注册表支持去中心化查找，但朴素的标识符哈希会通过枚举泄露成员资格。我们提出 VA-DAR，这是一种面向基于 ACE-GF 的钱包的带密钥的发现协议：此类钱包使用设备绑定通行密钥进行日常本地解锁，同时仅凭用户提供的标识符（例如电子邮件）和单一恢复口令（即 主口令）即可支持跨设备恢复。作为 ACE-GF 之上的发现与恢复层，VA-DAR 继承了 ACE-GF 上下文隔离、算法敏捷的派生基底，从而能够在身份层无中断地迁移至后量子算法。

该设计引入一个去中心化发现与恢复层，将保护隐私的发现标识符映射到存储于去中心化存储网络上的密封备份工件的不可变内容标识符。具体而言，用户使用内存困难型密钥派生函数派生以口令为根的密钥材料，对用于工件密封与发现索引的密钥进行域分离，并发布一条以由口令派生的发现标识符为键的注册表记录。

VA-DAR 提供：(i) 仅凭标识符和口令即可实用地实现跨设备恢复；(ii) 对公共目录枚举具备计算抗性；(iii) 通过所有者授权保障发现映射的完整性；以及 (iv) 通过单调版本控制和工件承诺实现回滚/篡改检测。我们定义了三种密封工件角色、两种更新授权选项和三种协议流程（注册、恢复、更新）。我们通过密码学博弈形式化定义安全目标，并证明在标准假设（内存困难型密钥派生函数的困难性、带密钥函数的伪随机性、签名不可伪造性和哈希抗碰撞性）下，VA-DAR 能够实现这些目标，同时保持供应商无关且不依赖特定区块链。要实现端到端后量子部署，还需要采用后量子安全的注册表授权实例化方案（选项 B 签名）。

1 引言

消费者钱包传统上依赖助记词种子短语进行恢复，这会导致可用性问题和托管失效。与此同时，通行密钥（硬件支持的凭证）可提供强大的本地身份验证，但通常与厂商同步或设备绑定状态相关联，这可能与严格的“无中心化服务器”要求相冲突。

本文提出了一种务实的混合方法：

- **本地安全性与便利性**。设备绑定的通行密钥伪随机函数（由硬件控制）会封存本地制品，以便在给定设备上通过生物识别解锁。
- **无需中心化服务器的跨设备恢复**。 备份制品仅由用户自行选择的恢复口令封存，并存储在去中心化存储中。公共注册表将一个带密钥的 发现标识符映射到制品的内容标识符。

核心贡献是 VA-DAR：一个发现与恢复层，它要求用户掌握恢复口令，才能计算用于注册表查找的发现标识符，从而避免公开枚举。

非目标。 VA-DAR 本身不会撤销已经派生的链上密钥，也不会使此前的地址失效。密钥撤销需要通过账户迁移或智能账户轮换机制实现，这不在本文档的范围内。

贡献

- 一种适用于公共注册表、能够抵御仅凭标识符枚举的密钥化发现构造。
- 一种域分离密钥派生方案，可防止发现、密封和注册表授权之间发生跨角色密钥重用。
- 两种更新授权选项（基于 HMAC 和基于签名），并对具体权衡进行分析。
- 一种带版本控制、承诺绑定的注册表更新协议，具备所有者授权和回滚抵抗性。
- 针对枚举抵抗、映射完整性和回滚安全性的形式化安全性游戏与定理级保证。

本研　　究基于标准原语构建了协议组合和安全模型；　　it
并未　　声称提出了新的底层密码学原语。

与先前 ACE-GF 工作的关系

ACE-GF[1] 被视为一项公开可用的原语规范，提供密封工件处理与确定性派生接口。本论文贡献了去中心化发现与恢复层、带密钥的发现标识符设计，以及注册表完整性与回滚模型。它并不改变 ACE-GF 的内部实现。

与相邻方法的定位对比

目前部署最广泛的恢复基线仍是助记词备份（例如 BIP-39）[9], ，它虽便于移植，但对主流用户而言在操作上较为脆弱。通行密钥/WebAuthn 改善了本地身份验证用户体验与抗网络钓鱼能力，但其核心标准范围是由身份验证器支持的登录，其本身并不是一套完整的去中心化钱包恢复协议 [10]。账户抽象生态系统（例如 ERC-4337 风格的智能账户）提供了强大的账户层控制能力，例如密钥轮换与恢复策略，但它们并不能直接解决无服务器备份发现中标识符查找所面临的公共目录枚举风险[11]。VA-DAR 被定位为一个互补层，专注于口令门控发现以及承诺与版本绑定的备份解析。

2 系统模型与预备知识

2.1 实体

- **客户端设备** D_i：运行钱包应用程序。
- **通行密钥伪随机函数** 伪随机函数$D_i(\cdot)$：一种与设备绑定的伪随机函数，须经本地用户在场验证或生物特征验证后方可使用（例如通过平台伪随机函数扩展）。
- **去中心化存储**：内容寻址存储，例如 Arweave [7] ，可提供不可变的内容标识符。
- **公共注册表**：一个公开、仅追加或可更新的映射层（例如，二层网络智能合约），用于存储发现记录。

- **中继器**（可选）：一个便捷组件，代表客户端提交注册表交易。

2.2 秘密与制品

- REV（根实体值）：钱包用于派生密码身份和密钥的根秘密。

- **主口令** P：用户已知的恢复口令（可移植秘密）。
- SA（密封工件）：使用派生的密封密钥加密 REV 的二进制工件。

2.3 密码学原语

我们假设：
- 内存困难型密码密钥派生函数：Argon2id(·) [2]。
- 用于域分离的密钥派生函数：HKDF(·) [4]。
- 消息认证：HMAC(·) [5]。
- 认证加密：AEAD.Enc/AEAD.Dec，关联数据为 [6]。
- 数字签名：Sign/Verify（EUF-CMA 安全） [8]。
- 抗碰撞哈希函数：$H(\cdot)$。

2.4 记号

- $\|$ 表示拼接。
- Norm(email) 表示对标识符进行规范化（转换为小写、去除空格等）。

- Zeroize(·) 表示安全清除内存。

3 威胁模型与安全目标

3.1 对手能力

对手 $\mathcal{A}$ 是概率多项式时间（PPT）的，且可以：
- 观察完整的公共注册表状态和所有链上数据。
- 使用任意标识符，不限次数地查询注册表。
- 如果从去中心化存储中获取密封制品（SA2），则尝试进行离线猜测攻击。

- 尝试抢先交易或覆盖注册表记录，但须遵守注册表授权规则。
- 提交相互竞争的交易、重放过期更新，并篡改存储载荷传递。

- 控制中继器和网络调度。
- 通过发送垃圾查询或拒绝存储/固定内容来发动拒绝服务攻击。

我们不 假设存在可信的中心化服务器来执行速率限制、身份验证或托管内容。除非通过在线/离线猜测，对手并不知道诚实用户的口令，也无法破解底层密码学假设。我们也不假设注册表部署具有全局唯一性：对手可以部署一个平行注册表，并复制公开元数据（包括 app_id）。因此，安全性要求客户端验证规范注册表_id，而不是依赖命名空间保密。

3.2 安全目标

G1：跨设备可恢复性。合法用户只需使用 $(I,\ P)$ 和公开可用的基础设施，即可在新设备上恢复 REV。

G2：抗枚举性。观察者不应仅凭知道某个标识符（例如电子邮件）就能获知钱包成员身份；注册表不应是公开目录。

G3：发现映射完整性。未授权方不能更新受害者的映射，使其指向攻击者选择的内容。

G4：回滚抵抗性。系统应能抵御在未经明确授权的情况下，将映射回退到较旧的内容标识符。

G5：最小化设备锁定。设备绑定伪随机函数可增强本地安全性，但跨设备恢复不能依赖它。

4 设计概述：三种密封制品

VA-DAR 对钱包生命周期中的三种制品角色进行了形式化定义：

制品	密封因子	用途
SA1（本地）	基于伪随机函数D_1派生	在设备 D_1 上每日解锁（生物特征-识别）。
SA2（备份）	主口令 P 仅	跨设备恢复（可移植）。
SA3（新的本地项）	由伪随机函数D_2派生	在新设备上每日解锁 D_2。

表1：密封工件角色。

关键性质在于，SA2**必须无需任何设备绑定伪随机函数即可解密**，否则在“无供应商同步”的约束下，更换设备将导致恢复失败。这种分离直接支持目标G5。

5 严格域分离的密钥派生

5.1 两阶段口令派生

为使查找能在获取 SA2之前进行，VA-DAR 会从同一个口令 P 派生出两个不同的值。

阶段 A（查找/认证根；仅可由$(I,\ P)$ **计算得出）**。定义一个确定性的查找盐：

$$\mathsf{salt}_{\mathrm{lookup}} \leftarrow H\big(\texttt{"va-dar:lookup:v1"} \,\|\, \mathsf{Norm}(I)\big). \tag{1}$$

然后推导：

$$K_{\mathrm{lookup}} \leftarrow \mathsf{Argon2id}(P, \mathsf{salt}_{\mathrm{lookup}}, \mathsf{params}_{\mathrm{lookup}}). \tag{2}$$

阶段 B（工件解密根；绑定到工件本地盐）。获取 SA2 后，从制品中解析 盐$_{pw}$，参数$_{pw}$ 并推导：

$$K_{\mathrm{sealroot}} \leftarrow \mathsf{Argon2id}(P, \mathsf{salt}_{pw}, \mathsf{params}_{pw}). \tag{3}$$

5.2 用于封装、索引和授权的独立密钥

为确保单一口令 P 在各协议角色（工件密封、发现索引、注册表授权）中的安全复用，客户端派生独立密钥：

$$K_{\mathrm{sa}} \leftarrow \mathsf{HKDF}(K_{\mathrm{sealroot}}, \mathsf{info} = \texttt{"acegf:sa2:seal"}), \tag{4}$$

$$K_{\mathrm{idx}} \leftarrow \mathsf{HKDF}(K_{\mathrm{lookup}}, \mathsf{info} = \texttt{"va-dar:discovery:index"}), \tag{5}$$

$$K_{\mathrm{reg}} \leftarrow \mathsf{HKDF}(K_{\mathrm{lookup}}, \mathsf{info} = \texttt{"va-dar:registry:auth"}). \tag{6}$$

此域分离具有规范性。实现不得将同一密钥直接同时用于AEAD封装以及注册表索引或授权。

6 发现标识符构造

6.1 密钥化发现标识符

设 I 为用户提供的标识符，例如电子邮件地址。设 $\mathsf{ctx}_{\mathrm{did}}$ 为公开的域分离上下文（例如应用程序标识符、注册表标识符、协议版本）。定义：

$$\mathsf{DiscoveryID} \leftarrow \mathsf{HMAC}(K_{\mathrm{idx}}, \mathsf{ctx}_{\mathrm{did}} \,\|\, \mathsf{Norm}(I)). \tag{7}$$

理由。与 H（I ‖全局_盐）不同，密钥化构造可防止第三方在不知晓 P 的情况下计算 DiscoveryID。这直接支持枚举抵抗（G2）。

6.2 长度与编码

DiscoveryID 是长度为 HMAC 输出长度的二进制字符串。用于注册表时，可将其编码为十六进制或 Base64 URL 编码。规范化函数规范化与上下文 $\mathsf{ctx}_{\mathrm{did}}$ 必须是确定性的并予以规定；不一致的设置会导致账户碎片化。实现可以为 $\mathsf{ctx}_{\mathrm{did}}$固定一种结构，例如 app_id ‖ registry_id ‖ version。在规范性要求上，对于链上注册表，registry_id 应至少包含 (chain_id,contract_address)。app_id 是公开的，并非 真实性秘密；真实性由客户端验证的规范 registry_id 锚定。

7 密封制品格式

7.1 备份制品 SA2 （密码密封）

SA2 必须包含解密和验证所需的全部参数，口令除外。最小格式如下：

字段	描述
`version`	制品格式版本
`salt_pw`	Argon2id 的盐
`params_pw`	Argon2id 参数（m、 t、 p）
`aead_alg`	AEAD 算法标识符
`nonce`	AEAD 随机数
`aad`	关联数据（上下文、元数据）
`ct`	AEAD。Enc(K_{sa}，随机数，REV，关联数据)（随机数 from 上一行）

表 2：SA2 记录格式。

7.2 本地工件SA1/SA3 **(PRF 密封)**

本地制品通过设备绑定的 PRF 派生密钥进行密封：

$$K_{\mathrm{local}} \leftarrow \mathsf{HKDF}\big(\mathrm{PRF}_{D_i}(\mathsf{label}),\, \mathsf{info} = \texttt{"acegf:local:seal"}\big). \tag{8}$$

具体标签由实现定义，但对于设备上的每个应用和每位用户都应保持稳定。跨设备恢复不得依赖本地工件（G5）。

8 注册表记录与更新授权

8.1 注册表记录结构

公共注册表存储从 DiscoveryID 到记录的映射：

字段	描述
`cid`	去中心化存储中 SA2 的内容标识符- age
`ver`	单调版本计数器
`commit`	对制品的承诺，例如 H(SA2)
`pk_owner`	首次注册时绑定的所有者验证密钥 tion
`auth`	更新授权证明（见下文）

表3：注册表记录结构。

8.2 更新授权选项

由于 不存在中心化服务器，因此更新授权必须可通过密码学手段验证 e.

选项 A：基于口令派生的 HMAC 授权。根据公式（6）派生更新密钥 K_{reg}，然后定义：

$$\mathsf{auth} \leftarrow \mathsf{HMAC}\big(K_{\mathrm{reg}},\, \mathsf{DiscoveryID} \,\|\, \mathtt{cid} \,\|\, \mathtt{ver} \,\|\, \mathsf{commit}\big). \tag{9}$$

验证需要使用 K_{reg} 重新计算 HMAC，即需要知道作为秘密的 K_{reg}；因此，除非通过提交-揭示方案、零知识证明或链下验证来实现，否则纯链上验证并不简单。选项 A 最适合支持隐私保护验证（例如零知识有效性证明）或链下验证器的注册表。

选项 B：存储在注册表中的所有者公钥。在首次注册时将所有者公钥 $\mathsf{pk}_{\mathrm{owner}}$ 存储在记录中，并确保首次写入密钥绑定具有原子性（对 (DiscoveryID，$\mathsf{pk}_{\mathrm{owner}}$) 采用仅在为空时初始化语义）。更新必须经过签名：

$$\sigma \leftarrow \mathsf{Sign}\big(\mathsf{sk}_{\mathrm{owner}},\, \mathsf{DiscoveryID} \,\|\, \mathtt{cid} \,\|\, \mathtt{ver} \,\|\, \mathsf{commit}\big). \tag{10}$$

链上验证非常直接。其缺点在于所有者密钥生命周期管理。常见的部署模式有两种：(i) 使用安全备份保护的随机生成的所有者密钥；或 (ii) 从口令派生材料进行确定性派生（用户体验更佳，但安全性还取决于口令猜测成本和熵）。

备注 1。若要立即部署，选项 B 通常更容易在二层智能合约上实现。在本文中，选项 B 是主要可部署实现。选项 A 则作为扩展进行分析，此外还假设验证器模型能够在不泄露秘密材料的情况下验证基于 *HMAC* 的授权。

8.3 回滚抵抗性

注册表必须确保每个 DiscoveryID 的 ver 单调递增（例如，仅接受满足 $\mathrm{ver}_{\mathrm{new}} > \mathrm{ver}_{\mathrm{old}}$ 的更新）。在没有密钥泄露的情况下，这可以防止回滚到较旧的 CID。

9 协议流程

9.1 流程 1：初始注册（设备 D_1）

1. 设备 D_1 持有 REV 和通过 PRF$_{D_1}$封装的本地 SA1。2. 用户选择主口令 P。3. D_1 通过通行密钥生物识别解锁 SA1，并在内存中重建 REV。4. D_1 通过公式 (1) 从 I获取盐值 $_{\text{lookup}}$，然后从 (I、 P) 派生 K_{lookup}（公式 (2)），并派生 K_{idx}、 K_{rego}。5. D_1 采样新的盐值 $_{pw}$ 并计算 K_{sealroot} （公式 (3)），然后派生 K_{sa}。6. D_1 通过使用 K_{sa}封装 REV 来构建 SA2。7. D_1 将 SA2 上传至去中心化存储，获得 CID。8. D_1 计算 DiscoveryID $=$H MAC(K_{idx}、ctx$_{\text{did}}$ ‖Norm(I))。9. D_1 写入注册表记录：DiscoveryID $\mapsto$ {cid、ver $=1$、commit、pk$_{\text{owner}}$、auth}，并在首次写入时以原子方式绑定 pk$_{\text{owner}}$ 。10. D_1 对内存中的 REV 和所有派生密钥调用安全清除。

9.2 流程 2：恢复（设备 D_2）

1. 用户在 D_2 上安装钱包，输入标识符 I 和主口令 P。2. D_2 通过公式（1）从 I获取盐值$_{\text{lookup}}$，然后从（I， P）派生 K_{lookup}（公式（2））， 并派生 K_{idx}。3. D_2 计算 DiscoveryID $=$HMAC(K_{idx}，ctx$_{\text{did}}$ ‖规范化（I））。4. D_2 向注册表查询 DiscoveryID，获取 CID 和记录元数据。5. D_2 使用 CID 从去中心化存储中获取 SA2。6. D_2 验证 H（SA2）$=$ 提交。7. D_2 从 SA2 解析（盐值$_{pw}$，参数$_{pw}$），派生 K_{sealroot}，然后派生 K_{sa}。8. D_2 使用 K_{sa}解密 SA2，并在内存中重构 REV。9. D_2 启用通行密钥，并在 PRF$_{D_2}$下封存新的本地 SA3。10. D_2 根据需要从 REV 派生链专属密钥和地址。11. 在密钥派生完成后， D_2 对 REV 调用安全清除。

9.3 流程 3：备份更新 / 发现撤销

此流程会将注册表映射更新为新的内容标识符（例如，在怀疑信息已泄露、口令轮换或定期安全维护之后）。它提供的是发现撤销（目录更新），而不是使先前派生的链上密钥在密码学上失效。

1. 用户在授权设备上通过通行密钥解锁本地制品（SA1 或 SA3），并在内存中重建 REV。2. 用户选择目标密码短语 P' （保持不变 $P'=P$ 或进行轮换 $P'\neq P$）。 3. 构建新的备份制品 SA2′ （使用新盐值$'_{pw}$），并用 P'封装后上传，获得内容标识符′。 4. **情形 1：口令保持不变**（$P'=P$）。DiscoveryID 保持不变；使用相同的密钥提交授权更新，版本为$\leftarrow$ ver $+1$。 5. **情况 2：口令轮换**（$P'\neq P$）。根据（I, P'）计算 DiscoveryID′，并在键 DiscoveryID′ 处创建新的注册表条目（其版本序列独立，从 1 开始）。 6. 在情况 2 中，将先前的 DiscoveryID 条目标记为墓碑化（或设置可选的重定向指针，指向 DiscoveryID′），并附上明确的迁移元数据和由策略定义的宽限期。 7. 从内存中安全清除 REV。

注释 2。 版本单调性是按注册表键强制执行的。当 P 发生变化时，DiscoveryID也会变化，因此连续性必须建模为键迁移工作流（旧键墓碑/重定向），而不是在旧键上进行原地版本递增。

10 形式化安全定义

我们将第3节中的安全目标形式化为密码学博弈。在所有博弈中，对手 $\mathcal{A}$ 都是概率多项式时间的。

定义 1 (枚举抵抗博弈 $\mathbf{G}_{\mathrm{enum}}$)。 挑战者采样一个比特 $b \leftarrow \{0,1\}$。若 $b=1$，挑战者将目标标识符 I^* 注册为一个口令，该口令从最小熵为 $H_\infty(P) \geq \mu$的分布中采样，生成 DiscoveryID$^* =$ HMAC(K^*_{idx}, $\mathsf{ctx}_{\mathrm{did}} \,\|\, 规范化(I^*)$)。若 $b=0$，则从输出空间中均匀随机抽取 DiscoveryID* 。 $\mathcal{A}$ 获得公共注册表访问权限（若 $b=1$，则包括 DiscoveryID* ）以及候选标识符集合，但不会 获得口令。 $\mathcal{A}$ 输出一个猜测 b'。若 $b'=b$ ，且其优势超过内存困难型口令猜测的暴力破解基准，则 $\mathcal{A}$ 获胜。

定义 2 (映射完整性博弈 $\mathbf{G}^{\mathrm{map}}$)。 挑战者为 DiscoveryID*注册一条记录，并提供授权材料 auth*（在主模型中，使用 $\mathsf{pk}^*_{\mathrm{owner}}$ 对应的签名）。该博弈包含首次注册安全性：如果 DiscoveryID* 最初为空，则只有能够原子性地绑定 $\mathsf{pk}^*_{\mathrm{owner}}$ 的授权初始化操作才会被接受。假设注册表提供原子条件写入（仅当为空时初始化），并具有标准的账本排序保证。$\mathcal{A}$ 获得完整的注册表访问权限，并可对任意DiscoveryID $\neq$DiscoveryID*发出任意创建/更新/删除请求。若 $\mathcal{A}$ 导致注册表接受针对 DiscoveryID* 的状态变更操作（创建、覆盖、更新或删除），却未使用受害者的密钥材料生成有效授权，则 $\mathcal{A}$ 获胜。

定义3 （回滚安全博弈 $\mathbf{G}_{\mathrm{roll}}$）。 挑战者为DiscoveryID*注册并更新记录，版本依次为1、2、...、...、 n，对应的内容标识符为 c_1、 c_2、.....、 c_n，承诺为h_1、.....、 h_n。 $\mathcal{A}$控制网络和存储层。 $\mathcal{A}$可以延迟或审查消息，但假设客户端查询的是已最终确定/最新的注册表视图（标准账本最终性假设）。$\mathcal{A}$ 获胜的条件是：客户端在对版本 n 执行恢复时，接受了对应于版本$j < n$ 的制品（过时内容标识符重放），或接受了制品 SA2′，使得 H（SA2′） $\neq h_n$ （篡改），且客户端未检测到差异。

11 主要结果

定理1 （枚举抵抗）。 假设Argon2id 在配置成本（参数lookup）下是单向的，并且HMAC（K_{idx}、 $\cdot$） 是以通过HKDF从 K_{lookup} 派生的 K_{idx} 作为密钥时的伪随机函数。那么，对于参与$\mathbf{G}_{\mathrm{enum}}$的任意概率多项式时间敌手 $\mathcal{A}$，

$$\mathrm{Adv}^{\mathrm{enum}}_{\mathcal{A}} \;\leq\; \mathrm{Adv}^{\mathrm{ow}}_{\mathsf{Argon2id}} \;+\; \mathrm{Adv}^{\mathrm{prf}}_{\mathsf{HMAC}} \;+\; \mathsf{negl}(\lambda).$$

也就是说，仅凭标识符枚举可归约为口令猜测，其每次猜测都需要付出内存困难型成本，且口令熵参数为 μ。

证明概要。 若没有 P，对手就无法派生 K_{lookup} ，因而无法计算 K_{idx}。若没有 K_{idx}，根据 HMAC 的伪随机函数性质，输出 DiscoveryID $=$HMAC(K_{idx}, $\mathsf{ctx}_{\mathrm{did}}$ ||Norm(I^*)) 在计算上与均匀随机字符串不可区分。因此， $\mathcal{A}$ 无法区分 $b=1$ 和 $b=0$ ，除非猜测 P；每次猜测都需要以配置的内存困难型成本执行完整的 Argon2id 派生流程。其优势可分解为内存困难型密钥派生函数的单向性优势与 HMAC 的伪随机函数区分优势；在所述假设下，两者均可忽略。

□

定理 2 （映射完整性（主要实例化：选项 B））。 设注册表在$\mathsf{pk}_{\mathrm{owner}}$ 下采用 *EUF-CMA* 安全方案强制执行签名验证，并强制实施仅在为空时初始化的语义，以便在每个DiscoveryID首次注册时，以原子性方式绑定$\mathsf{pk}_{\mathrm{owner}}$ 。对于所有者密钥配置，考虑：

（*R*）随机密钥模型： $\mathsf{sk}_{\text{owner}}$ 均匀生成并安全存储。（*D*）口令派生模型： $\mathsf{sk}_{\text{owner}}$ 由口令派生密钥材料确定性地派生。

那么，对于任何参与$\mathbf{G}_{\text{map}}$的概率多项式时间对手 $\mathcal{A}$，

$$\mathrm{Adv}^{\text{map}}_{\mathcal{A}} \;\le\; \begin{cases} \mathrm{Adv}^{\text{euf-cma}}_{\mathsf{Sig}} + \mathsf{negl}(\lambda) & \textit{(R)}, \\ \mathrm{Adv}^{\text{euf-cma}}_{\mathsf{Sig}} + \mathrm{Adv}^{\text{ow}}_{\mathsf{Argon2id}} + \mathsf{negl}(\lambda) & \textit{(D)}. \end{cases}$$

概率多项式时间对手无法实施未授权映射变更。

证明概要。 对手必须在无法访问秘密密钥的情况下，使用 $\mathsf{sk}_{\text{owner}}$ 生成有效签名。在模型 (R) 中，这正是 EUF-CMA 伪造。在模型 (D) 中，对手还可以尝试猜测口令，以重建派生的所有者密钥材料；这会产生内存困难猜测项。因此，成功篡改仍需要伪造签名，或攻破口令／密钥派生函数的边界。根据 EUF-CMA 安全性（以及模型 (D) 中 Argon2id 的单向性），即使对手能够获取其他消息上的签名（例如，同一 DiscoveryID*此前观察到的更新授权，这些授权对应不同的 (cid、ver、commit) 元组），其成功概率仍可忽略不计。

□

备注 3 (扩展实例化：选项 A)。 如果某个具体验证器模型能够在不暴露秘密材料的情况下安全验证基于 *HMAC* 的授权（例如，使用带有绑定声明的可靠零知识有效性证明），那么可以根据 *Argon2id* 的单向性和 *HMAC* 伪随机函数安全性，推导出相应的映射完整性界。

定理 3 （回滚/篡改安全性：基于新鲜的已最终化读取）。 如果注册表验证器强制要求版本号严格递增，且客户端在恢复时验证 $H(\mathsf{SA2}) = \text{提交}$，同时读取新鲜的已最终化注册表状态，那么对于任意参与游戏$\mathbf{G}_{\text{roll}}$的概率多项式时间敌手 $\mathcal{A}$，

9 个概率多项式时间归约器 $\mathcal{B}$，使得

$$\mathrm{Adv}^{\text{roll}}_{\mathcal{A}} \;\le\; \mathrm{Adv}^{\text{cr}}_{H} \;+\; \mathrm{Adv}^{\text{map}}_{\mathcal{B}} \;+\; \varepsilon_{\text{fresh}} \;+\; \mathsf{negl}(\lambda),$$

其中， $\varepsilon_{\text{fresh}}$ 表示尽管部署采用了新鲜性策略，客户端仍读取未最终化或陈旧注册表视图的概率。除发生概率可忽略的哈希碰撞外，篡改都会被检测到；而陈旧重放则需要违反新鲜已最终化读取假设或伪造授权。

证明概要。版本回滚。 注册表会拒绝任何版本号不满足 $\mathrm{ver}_{\text{new}} \le \mathrm{ver}_{\text{old}}$ 的更新。因此，对手无法通过合法注册表写入将映射恢复为较早的版本。若要向客户端呈现陈旧记录，就必须控制客户端的注册表视图，使新鲜性/最终性假设失效，或者伪造降级授权；后者与定理 2 矛盾。这两种失败事件分别贡献 $\varepsilon_{\text{fresh}}$ 和 $\mathrm{Adv}^{\text{map}}_{\mathcal{B}}$，其中 $\mathcal{B}$ 是一种归约：只要接受了未授权降级写入，它就利用回滚对手破坏映射完整性。

制品篡改。 如果 $\mathcal{A}$ 替换 SA2′ ≠SA2，但保留相同的内容标识符或承诺，则客户端的承诺校验 $H(\mathsf{SA2}') = \mathsf{commit}$ 会失败，除非 $H(\mathsf{SA2}') = H(\mathsf{SA2})$，这需要 H 中发生碰撞。

□

推论 1 （可恢复性 — G1）。由于仅凭(I, P)即可计算查找密钥（式(2)），并且从嵌入的(盐值$_{pw}$,参数$_{pw}$)中获取SA2 后即可推导出密封密钥（表 *2*），因此，任何持有(I, P)的新设备都可以重新计算DiscoveryID、获取SA2、验证承诺并解密，从而恢复REV。无需设备绑定秘密，即满足 *G1*。

12 隐私注意事项

12.1 公开可观测性

注册表是公开的。即使使用密钥化发现标识符，蓄意的攻击者仍可能：

- 如果他们已经知道某位用户的 DiscoveryID，便可随时间观察其更新。
    - 在不同网络之间关联更新时间或内容标识符。实现应尽量减少元数据，并避免在 SA2 的关联数据字段中嵌入可识别信息。

12.2 标识符规范化泄露

规范化必须是确定性的，且不得嵌入额外语义。原始标识符绝不得存储在链上。仅发布密钥化发现标识符。

12.3 迁移指针可链接性

如果在口令轮换期间使用链上重定向指针（流程 3，情况 2），旧 DiscoveryID 与新 DiscoveryID′ 之间的关联将公开可见，使观察者能够将两个注册表条目关联起来，确认它们属于同一用户。实现时应权衡此举带来的隐私代价与迁移提示的用户体验收益；如果需要不可链接性，则应将旧条目墓碑化，而不设置前向指针。

12.4 离线猜测

如果对手获得 SA2 密文，便可能针对 P尝试离线密码猜测。这是基于密码的恢复固有的风险。应将 Argon2id 参数设置为较高的内存成本，同时适应移动设备性能预算。可选的次级秘密（例如次级口令）可以增强恢复能力。

13 可用性注意事项

13.1 去中心化存储

内容寻址存储并不保证持久性。为确保可恢复性，SA2 必须可用：

- 用户可以固定自己的 SA2。
- 系统可以支持多个存储后端（多个内容标识符/副本）。
- 钱包可以允许使用可选的第三方固定服务提供商，但需承认这会重新引入一个可替代的中心化组件。

13.2 注册表可用性

如果注册表位于二层网络上，宕机或审查可能会阻碍恢复。客户端应支持从多个 RPC 端点读取，并可以支持镜像注册表。

14 撤销语义

更新 DiscoveryID →cid（或轮换到新的条目）可以实现：

- **发现撤销：**新设备将检索最新的 SA2，而不是旧的 SA2。它不会自动实现：
- **链上密钥撤销：**先前派生的密钥仍可能控制现有地址上的资产。

要真正撤销密钥，需要进行账户迁移、智能账户密钥轮换，或将资产转移至新派生的地址。

15 实现说明

参数选择。 两阶段派生（第 5 节）要求每次注册或恢复执行两次 Argon2id 运算。为限制总延迟，移动设备上的 $\text{params}^{\text{lookup}}$ 可以设置得比 params_{pw}更轻：阶段 A 仅负责枚举抵抗（G2），而阶段 B 则直接保护 REV。只要组合成本仍高于攻击者愿意付出的单次猜测预算，阶段 A 较弱也是可以接受的。具体参数集可以参考 RFC 9106 [3] 或平台特定指南。作为部署基线，阶段 A 的内存配置不应低于约 64 MiB，并且在目标客户端类别上应至少达到 80–150 毫秒；阶段 B 的强度至少应与阶段 A 相当（通常更强）。在实际运维中，部署方应公布目标离线猜测成本下限（每次猜测所需的时间和内存），以便审计安全性声明。每个密钥对应的 AEAD 随机数必须唯一；可考虑在 SA2 中存储一个随机数。所有敏感值（REV、派生密钥）在使用后都必须安全清除。

Passkey PRF 用法。 PRF 输出绝不应导出。仅使用它来派生本地密封密钥。应视情况将本地工件存储在操作系统安全 enclave/密钥链中。

DoS 防护措施。 由于不存在中心化服务器，DoS 缓解措施有限。客户端可以缓存成功的注册表查询、使用多个 Arweave 网关，并支持多个冗余内容标识符。

规范注册表绑定。 没有任何机制能够阻止第三方部署平行注册表，或重复使用 app_id 等公开元数据。因此，部署方应发布签名配置（或客户端硬编码允许列表），以绑定规范的 (app_id, registry_id)，其中 registry_id 至少包含 (chain_id，contract_address)。客户端应拒绝针对此经身份验证的绑定之外的注册表进行恢复/注册。

PQC 就绪与迁移。 由于 VA-DAR 构建于 ACE-GF 之上，它通过显式上下文分离（包括算法标识符），在身份/密钥派生层继承了算法敏捷性。实际而言，经典和后量子密钥域可以在同一身份根下共存，无需重新注册。要实现完整的端到端 PQ 部署，还必须在选项 B 中使用后量子安全签名来实例化注册表授权层；本文的安全性分析仍基于通用的签名 EUF-CMA 假设。

16 复杂度与成本

客户端成本。 主要来自两次内存困难型派生（Argon2id，每个阶段一次）和一次 AEAD 操作。相比之下，其他所有操作（HKDF、HMAC、签名）都可以忽略不计。

注册表成本。 每次变更都需要固定大小的密钥查找和一次签名验证（选项 B）。对于选项 A，验证成本取决于验证器模型：链下验证器直接执行基于密钥的检查，而链上部署通常需要辅助机制（例如承诺-揭示或零知识有效性证明），因此不假定链上存在统一的“单次 HMAC 检查”成本。每位用户的存储为 $O(1)$。

数据可用性。 读取操作是公开的，且与供应商无关；写入中继器是可选的便利组件，不影响安全性。

17 局限性

- 弱口令会削弱 G2 和离线猜测抵抗能力；策略必须强制要求足够的熵和密钥派生函数计算成本。

- 对于完全基于口令的可移植性而言，口令丢失会造成灾难性后果。
- 可用性取决于注册表和存储的可达性；必须采用多端点回退机制。
- 如果所有者的签名密钥遭到泄露（选项 B），回滚抵抗性就会失效；这是其固有属性。

- 在模型 (D) 中，映射完整性和抗枚举能力都受限于绕过口令猜测的难度；一次口令泄露就会同时破坏这两个目标。

18 未来工作

- **无需泄露秘密的链上授权：**基于口令派生的授权零知识证明（选项 A）。

- **反关联机制：**私有信息检索（PIR）或基于不经意伪随机函数（OPRF）的注册表访问。

- **真正的密钥撤销：**与智能账户集成，以支持密钥轮换和社交恢复。
- **多标识符支持：**通过隐私保护关联映射电话/电子邮件别名。

19 结论

VA-DAR 实现了实用的去中心化恢复，同时让公共目录发现受到密钥保护并以口令为门槛。该协议将三种工件角色分开——本地工件（PRF 密封）、备份工件（密码密封）和恢复后的本地工件——使设备绑定安全增强日常使用，同时避免恢复过程产生供应商锁定。域分离密钥派生、密钥化发现标识符和承诺绑定的版本化记录，为枚举抵抗、映射完整性和回滚保护提供了明确的密码学接口。在标准假设下，VA-DAR 满足其安全目标，同时保持供应商无关和链无关。当注册表授权（选项 B）采用后量子安全签名实现时，它还保留了通往端到端后量子部署的清晰迁移路径。

致谢

本文基于 ACE-GF 工件模型，并在此基础上扩展出一个实用的去中心化发现与恢复层。

参考文献

[1] J. S. Wang。ACE-GF：原子密码实体生成框架。*arXiv*预印本 *arXiv:2511.20505*，2025。`https://arxiv.org/abs/2511.20505`。[2] A. Biryukov、D. Dinu 与 D. Khovratovich。Argon2：用于密码哈希及其他应用的内存困难函数。载于*IEEE EuroS&P* 研讨会，2016。[3] Argon2：基于密码的密钥派生函数。RFC 9106，IETF，2021。

[4] H. Krawczyk 与 P. Eronen。基于 HMAC 的提取与扩展密钥派生函数（HKDF）。RFC 5869，2010年。[5] H. Krawczyk、M. Bellare 与 R. Canetti。HMAC：用于消息认证的密钥散列。RFC 2104，1997年。[6] P. Rogaway 与 T. Shrimpton。确定性认证加密：对密钥封装问题的可证明安全性处理。载于$EUROCRYPT$，2006年。[7] S. Williams 等。Arweave：一种用于永久信息存储的协议。Arweave白皮书，2023年。[8] S. Goldwasser、S. Micali 与 R. Rivest。一种可抵御自适应选择消息攻击的数字签名方案。*SIAM*计算杂志，17(2):281–308，1988年。[9] M. Palatinus、P. Rusnak、A. Voisine 与 S. Bowe。BIP-39：用于生成确定性密钥的助记码。比特币改进提案，2013年。 `https://github.com/bitcoin/bips/blob/master/bip-0039.mediawiki`。[10] 万维网联盟。Web身份验证：用于访问公钥凭证的应用程序编程接口（第2级）。W3C推荐标准，2021。 `https://www.w3.org/TR/webauthn-2/`。[11] V. Buterin、Y. Weiss、D. Kerekes 等。ERC-4337：使用替代内存池的账户抽象。以太坊改进提案，2023年。 `https://eips.ethereum.org/EIPS/eip-4337`。