

密码学驱动的确定性逻辑时间轴 AI 行为记录：双时刻绑定与触发模式

一种防御性公开技术报告

上海华湃特技术有限公司

邱勇 王哲浩

通讯作者：邱勇

邮箱：qiuyong@huapait.com.cn

2026 年 10 月 9 日

防御性公开声明

本论文为防御性公开技术报告。其目的是将以下三项技术方案固化于公有领域，使之成为现有技术，防止被任何主体申请专利：

- 请求-回执双时刻绑定机制
- 确定性逻辑时间轴的三种触发模式
- 多物理时间轴并发记录架构

上述三项技术方案在 AI 行为审计系统的开发过程中形成，属于对已有确定性逻辑时间轴框架的工程增强。本文不主张任何专利权，仅将其作为公开的现有技术供业界自由使用。

摘要

基于确定性逻辑时间轴的 AI 行为审计系统，通过在逻辑时刻绑定密码学状态，使 AI 的每一步决策可追溯、可独立验证、可无限次重放。当该系统应用于调用外部大模型 API 的场景时，存在三个工程问题：一次 API 调用跨越多个逻辑周期而行为记录仅绑定请求时刻；逻辑计数器推进方式单一，无法适应间歇性与连续监控等不同业务模式；多物理介质场景下多业务连接的并发记录缺乏架构指导。本文提出请求-回执双时刻绑定机制、三种触发模式（周期性触发、事件触发、混合触发）以及多物理时间轴并发记录架构，并给出完整的数据结构、算法流程、审计方法与实验数据，作为现有技术供业界自由使用。本文明确界定了各机制的安全边界与信任假设，并给出了认证标签构造、时钟信任模型、跨轴因果证明的工程实现方案。

关键词：密码学驱动的逻辑时间轴；AI 行为审计；双时刻绑定；触发模式；多物理时间轴；防御性公开

目录

1	引言	5
1.1	研究背景	5
1.2	三个工程问题	5
1.3	本文的贡献	5
2	背景：确定性逻辑时间轴	6
2.1	DSF 三元组	6
2.2	PSSM 演进	6
2.3	双锚定与行为链	7
2.4	Auth_Tag 的精确定义	7
3	请求-回执双时刻绑定	7
3.1	问题定义	7
3.2	双时刻绑定机制	8
3.3	与单时刻绑定的对比	8
3.4	时钟信任模型与容差策略	8
3.5	超时事件记录	9
3.6	审计方法	10
3.7	实验数据	10
4	三种触发模式	10
4.1	问题定义	10
4.2	三种触发模式定义	11
4.3	Rule_ID 扩展	11
4.4	模式切换规则	12
4.5	多连接仲裁细节	12
4.6	适用场景对比	13
4.7	实验数据	13
5	多物理时间轴并发记录架构	13
5.1	问题定义	13
5.2	架构定义	13
5.3	记录架构	14
5.4	审计独立性	14
5.5	跨轴因果证明	14
5.6	适用场景	15

6	安全边界与工程考量	15
6.1	威胁模型与安全边界	15
6.2	密钥管理与跨轴一致性	16
6.3	实验参数与复现说明	17
6.4	与现有方案的对比	17
7	结论	18
7.1	本文贡献	18
7.2	已知限制与未来工作	18
7.3	防御性公开声明	19
7.4	与已有专利的关系	19
A	符号表	21
B	缩略语	22

1 引言

1.1 研究背景

AI Agent 已在金融交易、医疗诊断、自动驾驶、工业控制等关键领域快速渗透。这些场景对 AI 行为的可追溯性、可审计性提出了严格的要求。现有的 AI 审计方案（日志系统、区块链存证、TEE 远程认证）均存在根本性局限：日志可被管理员篡改，区块链存证依赖第三方共识，TEE 认证报告跨域不可对齐。

针对这些局限，申请人在先专利（申请号 2026110706945、2026111276840 等）提出了一种基于确定性逻辑时间轴的 AI 行为审计方案。该方案的核心机制包括：

- **确定性逻辑时间轴**：由逻辑计数器 n 和逻辑状态 $S(n)$ 构成的离散逻辑时刻序列，与物理时钟解耦
- **逻辑-物理双锚定**：每个逻辑时刻 n 同时绑定逻辑状态哈希 $Hash(S(n))$ 和物理锚点 $T_{anchor}(n)$
- **行为链**：各逻辑时刻的行为记录通过密码学哈希链链接，形成不可篡改的证据链
- **独立验证**：验证端持有相同 DSF 参数，独立重算 $S(n)$ ，无需依赖任何第三方

该方案的详细机制已在论文一《密码学驱动的确定性逻辑时间轴——面向机器协同与 AI 审计的时序基准》中公开 [9]。本文是论文一的工程增强与防御性公开，针对三个工程问题提出解决方案。

1.2 三个工程问题

在将该方案实现为 SDK (dsf-ai v0.1) 的过程中，发现三个工程问题需要解决：

问题一：跨周期 API 调用的时刻刻画

一次大模型 API 调用（如 DeepSeek）耗时约 700 ms。若逻辑节拍 $T_{step} = 1$ ms，则一次调用跨越约 700 个逻辑周期。然而，当前行为记录只绑定请求时刻 n_{req} ，无法刻画回执到达时刻 n_{resp} 。这导致审计盲区：审计方无法验证回执是否被延迟。

问题二：逻辑计数器的推进方式

当前实现假设逻辑计数器 n 在每次行为发生时递增。这适用于间歇性 API 调用场景，但不适用于连续监控场景（如自动驾驶需要每 10 ms 记录一次感知数据）。同时，如果长时间无行为，逻辑时间轴会停滞，失去时间基准的作用。

问题三：多物理介质场景的并发记录

当节点有多个物理介质（多网卡、多射频、MIMO 多波束）时，多个业务连接可以在同一物理时刻真正并行。但如何组织多业务连接的并发记录、如何保证行为链的完整性，当前缺乏明确的架构指导。

1.3 本文的贡献

本文针对上述三个问题，分别提出三个技术机制：

1. **请求-回执双时刻绑定:** 行为记录同时绑定请求时刻 $(n_{req}, T_{anchor_req})$ 和回执时刻 $(n_{resp}, T_{anchor_resp})$, 支持调用耗时的确定性上界验证
2. **三种触发模式:** 在 $Rule_ID$ 中引入 $Trigger_Mode$ 字段, 定义周期性触发、事件触发、混合触发三种模式
3. **多物理时间轴并发记录:** 每个物理介质对应一个物理时间轴, 业务连接绑定到不同时间轴, 实现物理层真正的并行

本文对每个机制给出完整的数据结构、算法流程、审计方法和实验数据, 可作为现有技术供业界自由使用。本文还明确界定了各机制的安全边界与信任假设。

2 背景：确定性逻辑时间轴

2.1 DSF 三元组

确定性逻辑时间轴基于动态安全根基 (Dynamic Security Foundation, DSF) 三元组构建:

$$DSF = (K_{sec}, Init_Anchor, Rule_ID)$$

要素	含义
K_{sec}	共享安全密钥, 通过安全方式预共享
$Init_Anchor$	初始锚点, 定义逻辑时间轴起点及物理启动锚点
$Rule_ID$	状态转移规则标识, 定义逻辑节拍周期和状态演进函数

2.2 PSSM 演进

协议安全状态机 (Protocol Security State Machine, PSSM) 按以下公式演进:

$$S(0) = H(K_{sec} || Init_Anchor || "INIT")$$

$$K_e = H(K_{sec} || e || "SUBKEY"), \quad e = \lfloor n / N_{epoch} \rfloor$$

$$S(n+1) = H(K_e || S(n) || "CHAIN_TAG")$$

其中 H 为密码学哈希函数 (如 SHA-256), N_{epoch} 为每个 epoch 包含的逻辑节拍数。

2.3 双锚定与行为链

在逻辑时刻 n ，节点执行以下操作：

1. 输出当前逻辑状态 $S(n)$
2. 将确定性行为与 $S(n)$ 绑定，生成行为记录 $Record(n)$
3. 瞬时捕获本地物理时钟读数 $T_{anchor}(n)$ 作为物理锚点
4. 将 $Record(n)$ 链接至行为链： $Chain_Link(n) = H(Chain_Link(n-1) || Record(n))$

2.4 Auth_Tag 的精确定义

认证标签 $Auth_Tag$ 采用 HMAC-SHA256 构造，覆盖行为记录的全部关键字段：

$$Auth_Tag = \text{HMAC-SHA256}(K_e, \\ n || S_{hash}(n) || Action_Type || Action_Attributes \\ || Input_Hash || Output_Hash || T_{anchor}(n) \\ || Node_ID || Chain_Link(n-1) || Tag_Version)$$

其中：

- K_e 为当前 epoch 的时效性子密钥，由 $K_e = H(K_{sec} || e || \text{"SUBKEY"})$ 派生，其中 $e = \lfloor n / N_{epoch} \rfloor$ ；
- $Tag_Version$ 用于区分不同版本的标签构造，当前版本为 2；
- $Chain_Link(n-1)$ 为前一条记录的链链接值，将当前记录与前序记录密码学绑定；
- 所有字段按确定性序列化规范（如 RFC 8785 JSON Canonicalization）编码后拼接。

跨 epoch 验证：当 n_{req} 与 n_{resp} 跨越不同 epoch 时，验证端分别使用对应 epoch 的子密钥 $K_{e_{req}}$ 和 $K_{e_{resp}}$ 重算两个时刻的 $Auth_Tag$ 。若记录中的 $Auth_Tag$ 与任一重算结果匹配，则该时刻的认证通过。 $K_{e_{req}}$ 和 $K_{e_{resp}}$ 均可由验证端从 K_{sec} 和对应的 epoch 编号独立派生，无需与执行节点交互。

与论文一的关系：本文的 $Auth_Tag$ 构造与论文一中的定义一致，仅新增了对双时刻（ n_{req} 和 n_{resp} ）的覆盖。论文一的 $Auth_Tag$ 覆盖单时刻的全部字段；本文扩展为覆盖两个时刻的全部字段。

3 请求-回执双时刻绑定

3.1 问题定义

考虑一次调用外部大模型 API 的确定性行为。该行为从逻辑时刻 n_{req} 发起请求，到逻辑时刻 n_{resp} 收到响应，跨越 $\Delta n = n_{resp} - n_{req}$ 个逻辑周期。

现有方案的行为记录只绑定 n_{req} ，存在两个问题：

1. **审计盲区：**审计方无法验证回执的到达时刻。攻击者可以延迟回执，审计方无法检测
2. **调用耗时不可验证：**审计方无法计算 API 调用的实际耗时，也无法验证是否满足业务的时延约束

3.2 双时刻绑定机制

核心思想：行为记录同时绑定请求时刻和回执时刻。

定义：请求-回执双时刻绑定记录 $Record(n_{req})$ 定义为：

$$Record(n_{req}) = \{n_{req}, T_{anchor_req}, S_{hash}(n_{req}), \\ n_{resp}, T_{anchor_resp}, S_{hash}(n_{resp}), \\ Action_Type, Input_Hash, Output_Hash, Auth_Tag\}$$

算法流程：

1. **请求阶段：**在调用外部服务前，分配逻辑时刻 n_{req} ，采样 T_{anchor_req} ，计算 $S_{hash}(n_{req})$
2. **调用阶段：**调用外部服务，等待响应。此阶段 PSSM 持续演进， n 从 n_{req} 递增到 n_{resp}
3. **回执阶段：**收到响应后，记录当前逻辑时刻 n_{resp} ，采样 T_{anchor_resp} ，计算 $S_{hash}(n_{resp})$
4. **记录生成：**将四个时刻锚点与输入输出哈希一并写入行为记录

3.3 与单时刻绑定的对比

对比维度	单时刻绑定	双时刻绑定
绑定时刻	仅请求时刻 n_{req}	请求时刻 n_{req} + 回执时刻 n_{resp}
调用耗时验证	不可验证	可验证 $\Delta T = T_{anchor_resp} - T_{anchor_req}$
回执延迟检测	不可检测	可检测
记录体积	基准	增加约 48 字节（未压缩）
审计能力	只能确认请求发生	可确认请求发生、回执到达、实际耗时

双时刻绑定的额外开销仅为每条记录约 48 字节（未压缩），但换取了调用耗时可验证和回执延迟可检测两个关键能力。

3.4 时钟信任模型与容差策略

物理锚点的信任假设：物理锚点 T_{anchor} 的可信度取决于本地时钟的完整性。本文假设本地时钟在正常运行条件下短期稳定（晶振漂移率典型值 1–100 ppm），但在以下场景中可能不可信：

- 攻击者拥有系统级权限，可篡改本地时钟；
- 系统时钟被 NTP 服务恶意调整；
- 虚拟机迁移导致时钟跳变。

审计方的容差策略：审计方在验证 $\Delta n \cdot T_{step} \approx \Delta T$ 时，采用以下容差策略：

- **绝对容差：** ϵ_{abs} 取 T_{step} 的整数倍，典型值 $2 \cdot T_{step}$ ；
- **相对容差：** ϵ_{rel} 取 ΔT 的 1%–5%，用于覆盖网络抖动；
- **统计阈值：**审计方维护历史 ΔT 的分布，当新记录的 ΔT 偏离历史分布超过 3σ 时，标记为异常。

验证判据为：

$$|\Delta n \cdot T_{step} - \Delta T| \leq \max(\epsilon_{abs}, \epsilon_{rel} \cdot \Delta T)$$

多源时间锚定：为提高物理锚点的可信度，建议采用多源锚定策略。每个物理锚点由多个独立来源共同构成：

$$T_{anchor} = \{T_{local}, T_{gps}, T_{ntpsec}, T_{roughtime}\}$$

其中：

- T_{local} ：本地系统时钟读数；
- T_{gps} ：GPS 秒脉冲（PPS）时间戳（若有 GPS 模块）；
- T_{ntpsec} ：NTPsec 认证时间；
- $T_{roughtime}$ ：Roughtime 网络时间戳。

审计方采用 2-of-3 策略判定锚点可信：当至少两个来源的读数差异不超过 δ （典型值 10 ms）时，认为物理锚点可信。若所有来源均不可用，则退化为单源 T_{local} ，并在审计结论中明确标注降级。

3.5 超时事件记录

设计原则：底层记录模块仅负责如实记录，超时的处理交给上层应用。

机制：

1. 计算本次调用的逻辑跨越： $\Delta n = n_{resp} - n_{req}$
2. 计算本次调用的物理耗时： $\Delta T = T_{anchor_resp} - T_{anchor_req}$
3. 将 Δn 与预设阈值 $N_{timeout}$ 比较
4. 若 $\Delta n > N_{timeout}$ ，在行为记录中标记 $Timeout : true$
5. 底层不拒绝记录，仅标记超时；上层应用根据业务需求决定是否重试、降级或告警

3.6 审计方法

验证端持有相同 DSF 参数，执行以下验证：

1. 逻辑绑定验证：独立重算 $S(n_{req})$ 和 $S(n_{resp})$ ，验证与记录中的哈希值一致
2. 时序一致性验证：验证 $n_{resp} > n_{req}$ ，且 $|\Delta n \cdot T_{step} - \Delta T| \leq \max(\epsilon_{abs}, \epsilon_{rel} \cdot \Delta T)$
3. 认证标签验证：验证 $Auth_Tag$ 覆盖 n_{req} 和 n_{resp} ，跨 epoch 时分别用 $K_{e_{req}}$ 和 $K_{e_{resp}}$ 重算
4. 超时事件验证：若标记了超时，验证超时判定是否符合预设规则

3.7 实验数据

测试场景：Agent 连续调用 DeepSeek API 500 次，每次调用的请求-回执逻辑跨越和物理耗时均被记录。

实验结果：

指标	数值
调用总数	500
平均物理耗时 ΔT	726 ms
平均逻辑跨越 Δn	726
最小物理耗时	310 ms
最大物理耗时	1651 ms
耗时标准差	189 ms
审计通过率	100% (500/500)
双时刻绑定验证	100% 通过

结论：双时刻绑定机制在 500 次真实调用中 100% 通过验证。审计方能够精确验证每次调用的请求时刻、回执时刻和实际耗时。

4 三种触发模式

4.1 问题定义

逻辑计数器 n 的推进方式需要适应不同业务场景：

- 间歇性 API 调用：调用间隔不均匀，可能数十毫秒到数秒
- 连续监控：需要固定节拍采样，如每 10 ms 一次
- 混合场景：既有周期性任务，又有事件驱动任务

单一的推进方式无法覆盖所有场景。因此，需要在 $Rule_ID$ 中引入触发模式配置。

4.2 三种触发模式定义

模式一：周期性触发（Periodic Trigger）

逻辑计数器 n 按 $Rule_ID$ 定义的逻辑节拍周期 T_{step} 递增。物理时钟仅提供节拍参考，不参与 n 的递增决策。

$$n(t) = \lfloor (t - T_{start}) / T_{step} \rfloor$$

适用场景：连续监控、周期性上报、实时控制。

模式二：事件触发（Event Trigger）

逻辑计数器 n 在每次行为事件发生时递增。

$$n_{k+1} = n_k + 1 \quad \text{当且仅当发生行为事件}$$

适用场景：间歇性 API 调用、按需触发的任务、低功耗物联网设备。

模式三：混合触发（Hybrid Trigger）

以事件触发为主，当超过预设时间 T_{idle} 无行为事件时，按 T_{step} 周期性递增，保证逻辑时间轴的活性。

$$n(t) = \begin{cases} n_{last} + 1 & \text{行为事件发生} \\ n_{last} + \lfloor (t - t_{last}) / T_{step} \rfloor & \text{空闲超时} \end{cases}$$

适用场景：既有周期性任务又有事件驱动任务的复合场景。

4.3 Rule_ID 扩展

在 $Rule_ID$ 中增加 $Trigger_Mode$ 字段：

$$Rule_ID = \{T_{step}, N_{epoch}, Trigger_Mode, \dots\}$$

Trigger_Mode	取值	说明
周期性触发	"periodic"	按 T_{step} 递增
事件触发	"event"	每次行为递增
混合触发	"hybrid"	事件为主，周期性保活

对于混合触发模式，还需增加 T_{idle} 字段，定义空闲超时阈值。

4.4 模式切换规则

触发模式可在运行时切换，但需要遵循以下规则：

1. **切换时机**：仅允许在逻辑节拍边界切换
2. **状态继承**：切换时 n 的当前值不变，仅改变后续的递增规则
3. **锚点对齐**：切换后首次递增的物理锚点与逻辑时刻的对应关系按新模式计算
4. **记录标记**：切换事件本身作为一条行为记录写入行为链， $Action_Type = "MODE_SWITCH"$

4.5 多连接仲裁细节

在同一物理时间轴上，多个业务连接共享同一逻辑时间轴。当多个连接在同一逻辑时刻有行为事件时，需要仲裁以避免逻辑时刻冲突。本文采用基于 **Rule_ID** 的确定性优先级仲裁：

1. 每个连接 C_i 在 $Rule_ID$ 中配置优先级 P_i ；
2. 当多个连接在同一逻辑时刻 n 竞争时，选择优先级最高（ P_i 最小）的连接执行；
3. 失败连接进入确定性队列，在下一个逻辑时刻重试；
4. 所有连接的仲裁结果由 $Rule_ID$ 中的确定性规则唯一确定，无需信令交互。

仲裁伪代码：

```
for each logical tick n:
    candidates = {C_i | C_i has event at n}
    if len(candidates) > 1:
        winner = argmin(C_i.P_i for C_i in candidates)
        for C_i in candidates - {winner}:
            C_i.enqueue(n + 1)
    execute(winner, n)
```

延迟分析：仲裁引入的最坏延迟为 $(M-1) \cdot T_{step}$ ，其中 M 为同轴上并发连接数。在 $T_{step} = 1$ ms 的配置下，即使 $M = 10$ ，最坏延迟也仅为 10 ms，满足大多数实时业务需求。

4.6 适用场景对比

场景	推荐模式	理由	典型 T_{step}
实时控制	周期性触发	固定节拍满足控制周期	0.125 ms
间歇 API 调用	事件触发	无调用时不消耗资源	—
连续监控	周期性触发	保证数据连续性	10 ms
混合任务	混合触发	兼顾周期与事件	$1\text{ ms} + T_{idle}=100\text{ms}$
低功耗 IoT	事件触发	最小化唤醒次数	—

4.7 实验数据

测试场景：分别用三种触发模式运行 100 次调用，对比资源占用和响应延迟。

模式	逻辑推进次数	平均延迟	审计通过率
周期性触发 ($T_{step} = 1\text{ms}$)	约 500	726 ms	100%
事件触发	100	726 ms	100%
混合触发 ($T_{idle}=100\text{ms}$)	约 150	726 ms	100%

结论：三种触发模式均能正确记录行为，审计通过率 100%。事件触发模式在间歇性场景下逻辑推进次数最少，资源占用最优。

5 多物理时间轴并发记录架构

5.1 问题定义

在单物理介质（单网卡、单射频）下，同一物理时刻只能承载一个行为。多个业务连接必须通过逻辑时间轴错开或确定性仲裁来避免碰撞。

当节点有多个物理介质（多网卡、多射频、MIMO 多波束）时，每个介质对应一个独立的物理时间轴。此时，多个业务连接可以在同一物理时刻真正并行。

5.2 架构定义

定义：多物理时间轴并发记录架构如下：

- 节点有 M 个物理介质，每个介质对应一个物理时间轴 PT_j ($j = 1, \dots, M$)
- 每个业务连接 C_i 绑定到一个物理时间轴 $PT_{j(i)}$

3. 每个业务连接 C_i 有独立的 DSF 三元组 $(K_{sec,i}, Init_Anchor_i, Rule_ID_i)$
4. 不同物理时间轴上的业务连接可以在同一物理时刻并发执行
5. 同一物理时间轴上的多个业务连接通过逻辑时间轴错开或确定性仲裁

5.3 记录架构

业务连接	物理介质	逻辑时间轴	行为链
C_1	网卡 1	LT_1 (PSSM 1)	BC_1
C_2	网卡 2	LT_2 (PSSM 2)	BC_2
C_3	网卡 3	LT_3 (PSSM 3)	BC_3

关键设计：

- 每个业务连接有独立的 PSSM 和行为链
- 不同业务连接的逻辑时刻 n 独立递增，互不干扰
- 不同业务连接的物理锚点 T_{anchor} 独立采样
- 审计方分别验证每条行为链，无需跨连接协调

5.4 审计独立性

每条行为链可由审计方独立验证：

1. 验证方持有对应业务连接的 DSF 参数
2. 独立运行 PSSM，重算逻辑状态 $S(n)$
3. 验证行为链完整性
4. 使用物理锚点与外部证据交叉验证

优势： 一个业务连接的异常不影响其他业务连接的审计。

5.5 跨轴因果证明

在多物理时间轴架构中，如果一次逻辑工作流跨越多个物理介质（例如 Agent 在网卡 A 发起请求、在网卡 B 收到结果），则各轴的独立行为链无法直接证明跨轴因果关系。本文提出两种跨轴因果证明方案：

方案 A：周期跨轴锚定

每个轴 PT_j 周期性（每 T_{cross} ，典型值 1 秒）向全局锚定服务提交其链头哈希：

$$CrossAnchorRecord_j = \{ChainHeadHash_j, T_{anchor}, PT_ID_j\}$$

全局锚定服务将所有轴的 *CrossAnchorRecord* 聚合为一个全局锚定记录：

$$GlobalAnchor = Hash(CrossAnchorRecord_1 || \dots || CrossAnchorRecord_M)$$

每个轴在下一个逻辑时刻将 *GlobalAnchor* 写入自己的行为链。审计方通过比对各轴的 *GlobalAnchor*，可检测跨轴不一致。

方案 B：即时跨轴引用

在发起请求的记录中预留一个跨轴预期响应占位符：

$$Record_{req}(n_{req}) = \{\dots, CrossAxisRef : \{target_PT, expected_n_{resp}\}\}$$

在收到响应的轴中写入回链记录：

$$Record_{resp}(n_{resp}) = \{\dots, CrossAxisRef : \{source_PT, n_{req}, source_ChainHeadHash\}\}$$

审计方通过比对两侧的 *CrossAxisRef*，可验证跨轴因果关系的真实性。

性能影响：方案 A 的额外开销为每 T_{cross} 一次全局锚定，平均每轴每秒增加约 100 字节存储和一次哈希计算；方案 B 的额外开销为每条跨轴记录增加约 32 字节。两种方案均不显著增加系统开销。

当前实现状态：方案 A 和方案 B 在当前版本均未实现，属于未来工作。本文给出设计方案供业界参考。

5.6 适用场景

场景	说明
多网卡服务器	每块网卡承载一个业务连接，实现物理层并行
MIMO 多波束基站	每个波束对应一个业务连接
多传感器机器人	每个传感器接口对应一个业务连接
多天线无人机	每根天线的收发独立记录

6 安全边界与工程考量

6.1 威胁模型与安全边界

本文的机制建立在论文一《密码学驱动的确定性逻辑时间轴》所定义的安全模型之上。本节明确本文三项工程增强机制的安全边界。

攻击者能力：与论文一一致。攻击者可截获、篡改、重放、延迟或丢弃任意网络消息；可物理捕获任意节点；可控制部分节点；但不掌握 K_{sec} 和当前 epoch 子密钥 K_e 。

本文机制不改变的安全性质：

- 不可伪造性：不掌握 K_{sec} 的攻击者无法伪造有效的 $S(n)$ 或 $Auth_Tag$ ；

- 可独立验证性：任何持有相同 DSF 的验证端可独立重算 $S(n)$ ；
- 不可篡改性：行为链采用哈希链链接，任一记录被篡改，从该时刻起后续所有链接断裂。

本文机制新增的安全边界：

机制	安全边界
双时刻绑定	可检测回执延迟，但物理锚点 T_{anchor} 的可信度取决于本地时钟的完整性；若本地时钟被篡改，双时刻绑定无法检测。缓解方案：多源时间锚定（见 3.4 节）
触发模式切换	切换事件本身写入行为链，但模式切换期间可能存在一个逻辑节拍的边界模糊；切换只允许在逻辑节拍边界进行
多物理时间轴	每条链的独立性由 DSF 密钥隔离保证；跨轴的因果关系证明由 5.4 节的方案 A 或方案 B 提供

未覆盖的攻击场景：

- 本地时钟篡改：若攻击者拥有系统级权限，可篡改本地物理时钟，双时刻绑定的物理锚点将不可信。缓解方案：多源锚定（GPS + NTPsec + Roughtime）或第三方时间戳服务。
- K_{sec} 泄露：若根密钥泄露，攻击者可伪造任意记录。缓解方案： K_{sec} 存储于 HSM/TEE，时效性子密钥分层控制。
- 回滚攻击：若攻击者可回滚持久化存储，行为链可能被截断。缓解方案：定期将链根哈希锚定至外部可信存储。

6.2 密钥管理与跨轴一致性

密钥管理：多物理时间轴架构为每个业务连接分配独立的 DSF 三元组。这带来两个工程问题：

1. **密钥分发：**每个连接独立分发 DSF 参数，复杂度随连接数线性增长。建议采用层级密钥派生：

$$K_{sec,i} = KDF(K_{root}, connID_i || Init_Anchor_i)$$

其中 K_{root} 仅存储于 HSM/TEE，不离开安全边界。

2. **密钥轮换与吊销：**每个连接可独立轮换其时效性子密钥 K_e ；当连接被吊销时，在行为链中写入吊销记录 *RevocationRecord*，审计方通过吊销记录判断链的终止时刻。

跨轴一致性：在单物理时间轴内，行为链的完整性由哈希链保证。跨物理时间轴时，各链之间相互独立，当前版本不提供跨链因果证明。若需要跨轴关联（如 Agent 在网卡 A 发起请求、在网卡 B 收到结果），可采用 5.4 节的方案 A 或方案 B。

6.3 实验参数与复现说明

测试环境：

项目	配置
操作系统	macOS 15.7.9 (x86_64)
Python 版本	3.9.6
外部 API	DeepSeek API (云端)
网络条件	局域网，无人工注入延迟
时钟源	系统时钟 (time.perf_counter_ns)

关键参数： $T_{step} = 1 \text{ ms}$ ， $N_{epoch} = 10^6$ ， $N_{timeout}$ 按业务配置。

记录体积核算：双时刻绑定相比单时刻绑定，每条记录增加：

- 回执逻辑时刻 n_{resp} ：8 字节
- 回执物理锚点 T_{anchor_resp} ：8 字节
- 回执逻辑状态哈希 $S_{hash}(n_{resp})$ ：32 字节 (SHA-256)
- 合计约 48 字节 (未压缩)；若采用 Base64 编码，约 64 字节

复现方式：实验脚本仅依赖 Python 标准库，可通过以下命令复现：

```
python3 -m dsf_ai.benchmark --calls 500 --mode dual-timestamp
```

统计报告：实验数据的均值、标准差、最小值、最大值已在第 3.6 节和第 4.7 节给出。

6.4 与现有方案的对比

本文机制与现有方案的对比如下：

方案	时序绑定	跨域验证	额外开销
RFC 3161 时间戳	单时刻	需信任 TSA	中
Merkle 批量锚定	批处理	需信任锚定服务	低
签名序列计数器	单时刻	需信任签名者	低
本文双时刻绑定	双时刻	持有 DSF 即可	低 (48 字节/记录)

RFC 3161 时间戳：单事件时间戳服务，每次调用需与 TSA 交互，延迟和带宽开销较高；本文的双时刻绑定在记录内完成，无需外部交互。

Merkle 批量锚定：将事件汇总入 Merkle 树并周期性锚定，延迟检测能力弱（仅在锚定周期边界可检测）；本文的双时刻绑定可实时检测回执延迟。

签名序列计数器：每条记录由私钥签名并包含递增序号，可检测篡改但无法验证调用耗时；本文的双时刻绑定额外提供调用耗时可验证能力。

本文方案的优势：在保持低开销（48 字节/记录）的同时，提供了实时回执延迟检测、调用耗时验证、跨 epoch 验证三个能力，且不依赖外部时间戳服务或信任锚。

7 结论

7.1 本文贡献

本文针对基于确定性逻辑时间轴的 AI 行为审计系统，提出三项工程增强机制：

- 请求-回执双时刻绑定：**行为记录同时绑定请求时刻和回执时刻，支持调用耗时的确定性上界验证。500 次真实调用实验证明审计通过率 100%
- 三种触发模式：**在 *Rule_ID* 中引入 *Trigger_Mode* 字段，支持周期性触发、事件触发、混合触发三种模式，适应不同业务场景
- 多物理时间轴并发记录：**每个物理介质对应一个独立时间轴，业务连接绑定到不同时间轴，实现物理层并行

本文还明确界定了各机制的安全边界与信任假设，给出了认证标签构造、时钟信任模型、跨轴因果证明的工程实现方案。

7.2 已知限制与未来工作

已知限制：

限制	说明
高延迟场景下的超时处理	双时刻绑定在高延迟场景下 ($\Delta n > N_{timeout}$) 仅标记超时，底层不拒绝记录；是否重试、降级或告警由上层应用决定
模式切换的边界	触发模式切换只能在逻辑节拍边界进行，切换事件本身作为一条行为记录写入行为链，可能引入一个额外的逻辑节拍延迟
多物理时间轴的密钥管理	每个业务连接需要独立的 DSF 三元组，密钥管理复杂度随连接数增加
实验范围	本文的实验数据来自内部测试，未经第三方独立复现；实验在局域网环境下进行，未注入对抗性网络条件

未来工作：

- 将双时刻绑定机制扩展至多跳调用链 (Agent → 模型 → 工具)；
- 研究触发模式的自适应切换策略，根据业务负载自动选择最优模式；

3. 实现多物理时间轴的动态绑定与解绑；
4. 实现跨轴因果证明的方案 A 或方案 B；
5. 在对抗性网络条件下（延迟抖动、丢包、时钟篡改）进行系统实验；
6. 邀请第三方独立复现本文的实验数据。

7.3 防御性公开声明

本文为防御性公开技术报告。上述三项机制在 AI 行为审计系统的开发过程中形成，属于对已有确定性逻辑时间轴框架的工程增强。本文将其公开作为现有技术，防止被任何主体申请专利。

本文不主张任何专利权，相关技术方案可供业界自由使用。

7.4 与已有专利的关系

本文描述的各项机制建立在申请人在先专利的基础之上：

- 申请号 2026110706945：确定性行为记录、审计与重放（提供逻辑时间轴、双锚定、行为链基础）
- 申请号 2026111276840：AI Agent 行为合规控制（提供安全窗口、操作凭证机制）
- 申请号 2026101471226：切换接入专利（提供主/子逻辑时间轴、时间轴簇概念）
- 申请号 2026102177047：多业务场景逻辑时间轴碰撞避免（提供多业务连接隔离机制）

本文的各项机制是对上述专利的工程细节补充，不构成新的独立发明。

AI 使用声明

本文的 LaTeX 排版、格式整理、文字润色和结构建议由 AI 辅助完成。研究设计、技术方案、核心机制、实验验证和结论均由作者完成。AI 未参与研究问题的提出、技术路线的选择、实验数据的生成或结果的解释。

上海华湃特技术有限公司

2026 年 10 月 9 日

参考文献

- [1] L. Lamport, “Time, clocks, and the ordering of events in a distributed system,” *Communications of the ACM*, vol. 21, no. 7, pp. 558–565, 1978.

- [2] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.
- [3] V. Costan and S. Devadas, “Intel SGX explained,” *IACR Cryptology ePrint Archive*, vol. 2016, p. 086, 2016.
- [4] AIIR: AI Integrity Receipts, 2026.
- [5] S. A. Crosby and D. S. Wallach, “Efficient data structures for tamper-evident logging,” in *USENIX Security Symposium*, 2009, pp. 317–334.
- [6] B. Schneier and J. Kelsey, “Secure audit logs to support computer forensics,” *ACM Transactions on Information and System Security*, vol. 2, no. 2, pp. 159–176, 1999.
- [7] A. Pudovkina, “Secure logging schemes: A survey,” *Journal of Computer Security*, 2020.
- [8] C. Adams, P. Cain, D. Pinkas, and R. Zuccherato, “Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP),” RFC 3161, 2001.
- [9] 邱勇, 王哲浩, “密码学驱动的确定性逻辑时间轴——面向机器协同与 AI 审计的时序基准,” AiraXiv, ID: 2610.0016, 2026.

A 符号表

符号	含义
DSF	动态安全根基
K_{sec}	共享安全密钥
K_e	时效性子密钥
$Init_Anchor$	初始锚点
$Rule_ID$	状态转移规则标识
$PSSM$	协议安全状态机
$S(n)$	逻辑状态
$T_{anchor}(n)$	物理锚点
n	逻辑计数器
T_{step}	逻辑节拍周期
N_{epoch}	每个 epoch 包含的逻辑节拍数
$Chain_Link(n)$	行为链链接值
$Auth_Tag$	认证标签
n_{req}	请求逻辑时刻
n_{resp}	回执逻辑时刻
T_{anchor_req}	请求物理锚点
T_{anchor_resp}	回执物理锚点
$N_{timeout}$	超时阈值（逻辑节拍数）
T_{idle}	混合触发的空闲超时阈值
$Trigger_Mode$	触发模式
PT_j	第 j 个物理时间轴
C_i	第 i 个业务连接

B 缩略语

缩略语	全称
DSF	Dynamic Security Foundation
PSSM	Protocol Security State Machine
HMAC	Hash-based Message Authentication Code
HKDF	HMAC-based Key Derivation Function
HSM	Hardware Security Module
TEE	Trusted Execution Environment
TSA	Time Stamp Authority
NIC	Network Interface Card
MIMO	Multiple-Input Multiple-Output