

Cryptography-Driven Deterministic Logical Timeline

A Temporal Baseline for Machine Collaboration and AI Auditing

Shanghai Huapait Technology Co., Ltd.

Qiu Yong Wang Zhehao

Corresponding author: Qiu Yong

Email: qiuyong@huapait.com.cn

October 9 2026

Abstract

Mobile communication is transitioning from “human communication” to “machine communication”. The core scenarios of 6G—industrial control, satellite constellations, UAV swarms, and AI agents—impose entirely new requirements on communication: deterministic latency, intrinsic security, zero-signaling collaboration, and infrastructure-free self-organization. However, the underlying logic of existing communication systems has always been “first align physical clocks, then allocate resources through signaling interactions”, a paradigm that encounters fundamental bottlenecks in machine communication scenarios.

This paper proposes a **cryptography-driven deterministic logical timeline**, driven by a Dynamic Security Foundation (DSF) that operates a Protocol Security State Machine (PSSM). Each node independently evolves a logical state $S(n)$ and samples a physical anchor $T_{anchor}(n)$ at logical decision moments, forming a logical-physical dual anchoring. Behavior records are linked via cryptographic hash chains. A verifier holding the same DSF can independently recompute $S(n)$, verify the integrity of the behavior chain, and trace the physical time position.

This paper systematically elaborates on three fundamental dilemmas: the irreversibility of physical time, the untrustworthiness of physical clocks, and the inability to independently verify third-party stored evidence. It provides a security model, contrasts the proposed scheme with Lamport clocks, blockchain storage, TEE, and other approaches, and validates feasibility through an AI call auditing example. The logical timeline serves as a unified temporal baseline for AI compliance control, token generation determinism, output determinism, and cross-domain verification.

Keywords: cryptography-driven logical timeline; dynamic security foundation; protocol security state machine; logical-physical dual anchoring; AI behavior auditing; cryptographic temporal baseline

Note

This is the English translation of the paper originally published in Chinese on AiraXiv (ID: 2610.0016). The research content is identical.

1 Introduction

1.1 An Unrecognized Paradigm Shift

Mobile communication is undergoing a fundamental paradigm shift. Systems up to 5G primarily served “humans”—mobile Internet, video calls, mobile payments. The core service object of 6G is shifting to “machines”—industrial controllers, autonomous vehicles, UAV swarms, satellite constellations, and AI agents.

These two types of communication objects have requirements:

Requirement	Human Communication	Machine Communication
Latency	As low as possible, statistical	Must have mathematically provable bound
Reliability	99.9% acceptable	Above 99.9999%, no long tail
Synchronization	Depends on GPS/1588	Cannot rely on external timing
Signaling	Signaling interaction acceptable	Signaling overhead must approach zero
Security	External encryption acceptable	Intrinsic security, physical layer protection
Networking	Depends on base station infrastructure	Infrastructure-free self-organization
Mobility	Handover interruption of tens of ms acceptable	Zero interruption, UE unaware
Power	Acceptable	Extremely low power, long standby

The deeper implication of this shift has not been fully recognized: **the underlying logic of communication systems itself needs to be re-examined.** Existing systems follow an invariant underlying logic—first align physical clocks, then conduct communication interactions. The physical clock is the “heartbeat” of the system, and all timing calculations are bound to the physical frame number. This paradigm works well in human communication scenarios but encounters fundamental bottlenecks in machine communication scenarios.

1.2 Three Fundamental Dilemmas

Dilemma 1: Physical time is irreversible.

After an incident, the clock cannot be turned back, and the environment cannot be reproduced. Traditional approaches can only rely on “attestation”—recording the data at the time, trusting that the attestation has not been tampered with. But “trust” is not “proof”. Log records can only prove that “someone claims something happened”, not that “something actually happened”. Blockchain attestation can only prove that “data was written at a certain moment”, not that “the data was correct at that time”. The root of this dilemma is that the physical world cannot be replayed—we cannot go back to “that time” to re-verify whether the behavior was correctly executed.

Dilemma 2: Physical clocks are untrustworthy.

Even if physical time is unified through GNSS or network reference clocks, physical clocks still have inherent limitations: they can be tampered with, spoofed, cannot be independently verified across domains, and cannot be replayed. Specifically:

- Model signature schemes: Even with a timestamp, the timestamp comes from the local physical clock, can be tampered with, and the verifier cannot independently recompute;
- TEE schemes: The attestation report is a single-node product; across domains, the timestamps of different nodes cannot be directly aligned;
- ZKML schemes: They prove computational correctness, but the proof itself contains no time information and cannot answer when, which model, and by whom;
- Runtime receipt schemes: The timestamp in the receipt comes from the local physical clock, can be tampered with, and the receipt cannot trace back the model identity;
- Statistical detection schemes: If the signed object includes a timestamp, it is still a physical timestamp; if not, it cannot answer when.

Dilemma 3: Third-party attestation cannot be independently verified.

Existing schemes—GPS timestamps, NTP/PTP synchronization, blockchain attestation, TEE remote attestation—all implicitly assume that the verifier must trust some external entity: GPS satellites, time servers, blockchain consensus nodes, or hardware vendors. The auditor must trust the credibility of the attestation platform and cannot independently verify. In cross-institutional, cross-trust-domain, and cross-physical-clock-domain scenarios, these assumptions all fail.

1.3 Contributions

This paper proposes a **cryptography-driven deterministic logical timeline** as a unified temporal baseline for machine collaboration and AI auditing. The core contributions include:

1. **Problem definition:** Systematically elaborates on three fundamental dilemmas—irreversibility of physical time, untrustworthiness of physical clocks, and inability to independently verify third-party attestation—revealing the common root of failure of existing schemes in machine communication and AI auditing scenarios.
2. **Mechanism proposal:** Proposes a **cryptography-driven** Dynamic Security Foundation (DSF) triple, Protocol Security State Machine (PSSM), logical-physical dual anchoring, behavior chain cryptographic linking, and candidate-set verification, constructing a deterministic logical temporal baseline decoupled from physical clocks, independently verifiable, and infinitely replayable.
3. **Boundary delineation:** Systematically contrasts the proposed scheme with Lamport clocks, NTP/PTP, GPS, blockchain attestation, TEE, ZKML, and Inference Receipts, clarifying its unique position.
4. **Security model:** Provides formal definitions and analysis of security properties including unforgeability, independent verifiability, tamper-evidence, forward/backward security, and layered key exposure control.
5. **Minimal validation:** Uses AI call auditing as an example to demonstrate the complete closed loop of “logical state recomputation + behavior chain integrity + physical anchor traceability”, and validates both strict synchronization and bounded-deviation modes through a three-node engineering demonstration.

2 Related Work and Fundamental Limitations of Existing Technologies

2.1 Physical Time Baseline Schemes

GPS/GNSS/Atomic Clocks: Used in cellular networks, power systems, and financial trading systems. Each node obtains absolute physical time from satellite atomic clocks via a GPS receiver and timestamps behavior records before storing them in a log system. Fundamental limitations: dependence on continuous external timing signals, unavailable indoors, underground, underwater, in deep space, or under strong interference; GPS signals are susceptible to spoofing and jamming, with single-point failure risk; log systems are vulnerable to tampering by users with administrator privileges.

IEEE 1588 / NTP: Measures propagation delay through message exchange to compensate for clock offset. This scheme relies on a “master-slave architecture” and implicitly assumes that all nodes’ physical clocks follow the same measurement rules, essentially “correcting deviation” rather than “establishing a new baseline”; centralized attestation platforms have single-point tampering risks, and auditors must trust the attestation platform, unable to independently verify.

2.2 Logical Clock Schemes

Lamport Clocks: Proposed by Lamport in 1978, and subsequent vector clock schemes, used in distributed systems to establish partial ordering of events (“A happened before B”). Core characteristics: a monotonically increasing software counter completely unrelated to physical clocks, updated through event-driven and message-passing mechanisms, only establishing causal order between nodes. Fundamental limitations: does not record the content of behaviors, does not record inputs and outputs, does not produce any form of verifiable evidence chain, and does not have auditing or replay capabilities. Although some existing technologies use Lamport clocks for log merging or audit trail construction, they only utilize their “ordering” capability, neither binding physical anchors nor recording behavior context.

Hybrid Logical Clocks (HLC): Studies timestamp schemes under bounded deviation, addressing the causality and physical time proximity of timestamps, but does not involve cryptographic commitment verification.

2.3 Attestation and Auditing Schemes

Blockchain Attestation: The Civil Aviation Administration of China has issued “Technical Requirements for Blockchain-Based Flight Data Attestation of Civil Unmanned Aircraft” (MH/T 2015-2024), requiring flight data to be attested on-chain. Fundamental limitations: essentially “external attestation”—data is uploaded to a blockchain or centralized platform after generation, and its credibility depends on the honesty and consensus mechanism of consortium nodes; auditors must trust the attestation service provider and cannot independently verify data authenticity; blockchain attestation must be uploaded after the fact, cannot be bound in real time when the behavior occurs, and there is a tampering risk within the time window.

Inference Receipts / AI Integrity Receipts: Generates lightweight cryptographic commitment records after AI inference is completed, binding model identity, sampling configuration, and output tokens into a tamper-evident hash chain. Fundamental limitations: post-hoc verification rather than pre-hoc control—the receipt is generated after inference is completed and can only prove that “a result was produced at a certain time”, not that the inference process itself is deterministic; does not involve entropy source control; explicitly does not guarantee universal deterministic replay across hardware or quantization methods; the security model is based on the “honest emitter” assumption.

TEE (Trusted Execution Environment): Utilizes hardware capabilities such as Intel SGX and NVIDIA H100 confidential computing to run models in isolated environments and generate remote attestation reports. Fundamental limitations: depends on specific hardware, has side-channel attack risks; each verification only targets the currently loaded model, the attestation report is isolated, and cannot provide a unified, tamper-evident time baseline in a distributed system across nodes and clock domains to bind model identity and execution moment.

ZKML (Zero-Knowledge Machine Learning): Converts model inference into arithmetic circuits and generates zero-knowledge proofs. Fundamental limitations: extremely high computational overhead; generating zero-knowledge proofs for large models costs 1000 to 10000 times the original inference, currently only feasible for very small models or inference slices; can only prove computational correctness, cannot provide historical auditing of behavior patterns.

2.4 Common Dilemma: Physical Time is Irreversible

All existing schemes share a fundamental dilemma: **physical time is irreversible**.

After an incident, the clock cannot be turned back, and the environment cannot be reproduced. Traditional approaches can only rely on “attestation”—recording the data at the time, trusting that the attestation has not been tampered with. But “trust” is not “proof”. Log records can only prove that “someone claims something happened”, not that “something actually happened”. Blockchain attestation can only prove that “data was written at a certain moment”, not that “the data was correct at that time”.

The root of this dilemma is that **the physical world cannot be replayed**—we cannot go back to “that time” to re-verify whether the behavior was correctly executed.

3 Cryptography-Driven Deterministic Logical Timeline: Core Mechanisms

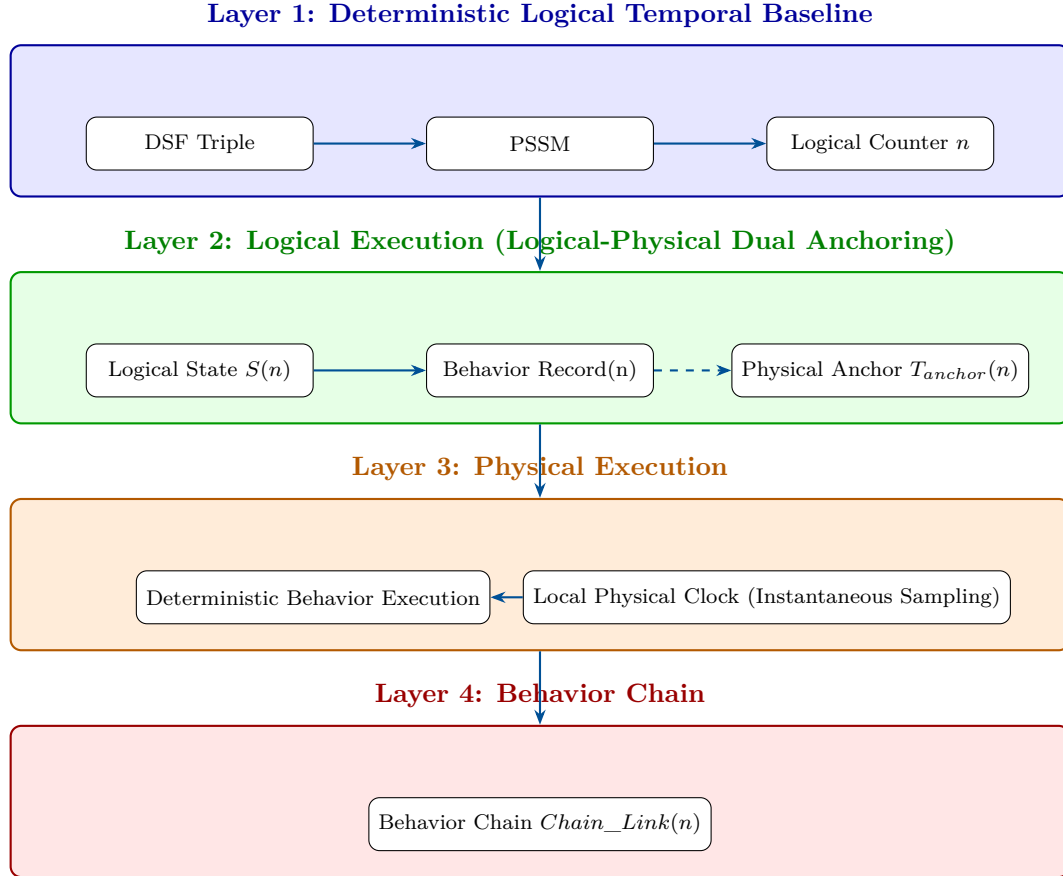


Figure 1: Four-layer architecture of the deterministic logical timeline and logical-physical dual anchoring design

3.1 Design Philosophy

The design philosophy of DSF can be summarized in three sentences:

What can be computed is not transmitted; what can be determined is not randomized; what can be logical is not physical.

Principle	Meaning	Subversion of Traditional Schemes
Compute, don't transmit	Any parameter computable from shared information is never transmitted over the air	Eliminates signaling overhead
Determine, don't randomize	Converts inevitable collisions into predictable queuing	Eliminates statistical long tail
Logical, not physical	Logical timeline decoupled from physical clock, using cryptographic determinism against physical imperfection	Eliminates physical clock dependence

3.2 From “Slave of the Physical Clock” to “Master of Logical Time”

Traditional communication systems follow an invariant underlying logic: **first align physical clocks, then conduct communication interactions**. The physical clock is the “heartbeat” of the system, and all timing calculations are bound to the physical frame number (SFN).

DSF breaks out of this paradigm, decomposing the time dimension of communication into two independent layers:

Timeline	Definition	Characteristics
Logical timeline	Defined by logical counter n and logical period T_{step} , driven by PSSM	Decoupled from physical clock, independently evolves according to cryptographic rules
Physical timeline	Defined by each node's local physical clock	Instantaneously sampled only at logical decision moments, used for physical execution

The relationship between the two timelines is **anchoring**, not **driving**. The physical clock provides a reference for the execution moment but does not define the content of the logical state.

Key insight: The physical clock is a tempo source, not a sync source. It determines “how often to step”, not “which step to take”. The content of the logical timeline is independently determined by **cryptographic rules**.

3.3 DSF Triple

DSF is defined by a triple:

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

Element	Meaning
K_{sec}	Shared secret key, 256 bits, pre-shared via a secure channel, stored in a hardware security module
$Init_Anchor$	Initial anchor, defining the logical timeline starting point and its corresponding physical start anchor
$Rule_ID$	State transition rule identifier, defining the logical tick period T_{step} , the number of logical ticks per epoch N_{epoch} , the state evolution function, and safety window parameters

3.4 Protocol Security State Machine (PSSM)

Both communicating parties independently run the PSSM based on the shared DSF:

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

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

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

where H is a cryptographic hash function (e.g., SHA-256, SHA-3, BLAKE2, SM3). The state $S(n)$ of the PSSM has the following properties:

Property	Description
Unforgeability	A third party not holding K_{sec} cannot predict or forge $S(n)$
Independent verifiability	Any node holding the same K_{sec} can independently recompute $S(n)$
Tamper-evidence	Once $S(n)$ is recorded, any modification will break the subsequent hash chain
Forward security	Even if $S(n)$ is compromised, $S(n+1)$ cannot be inferred
Backward security	Even if the device is physically captured, historical $S(k)$ ($k < n$) cannot be recovered

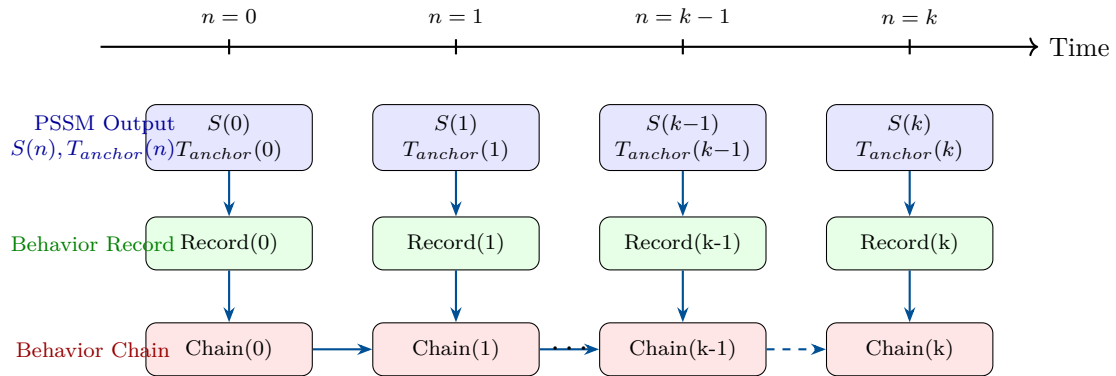


Figure 2: PSSM state evolution and logical-physical dual anchoring timing diagram

3.5 Logical-Physical Dual Anchoring

At logical moment n , the node performs the following operations:

1. Outputs the current logical state $S(n)$;
2. Binds the deterministic behavior to $S(n)$, generating a behavior record $Record(n)$;
3. Instantaneously captures the local physical clock reading $T_{anchor}(n)$ as the physical anchor, associating it with $Record(n)$;
4. Links $Record(n)$ to the behavior chain.

The behavior record $Record(n)$ contains at least the following information:

Field	Meaning	Purpose
n	Logical moment identifier	Determines the behavior's position in the logical sequence
$S(n)$ $Hash(S(n))$	Current logical state	Logical replay verification
$T_{anchor}(n)$	Physical anchor	Physical traceability
$Action_Type$	Behavior type identifier	Distinguishes behavior types
$Input_Hash$	Hash of input data	Logical replay
$Output_Hash$	Hash of output result	Logical replay
$Auth_Tag$	Authentication tag	Tamper-evidence verification

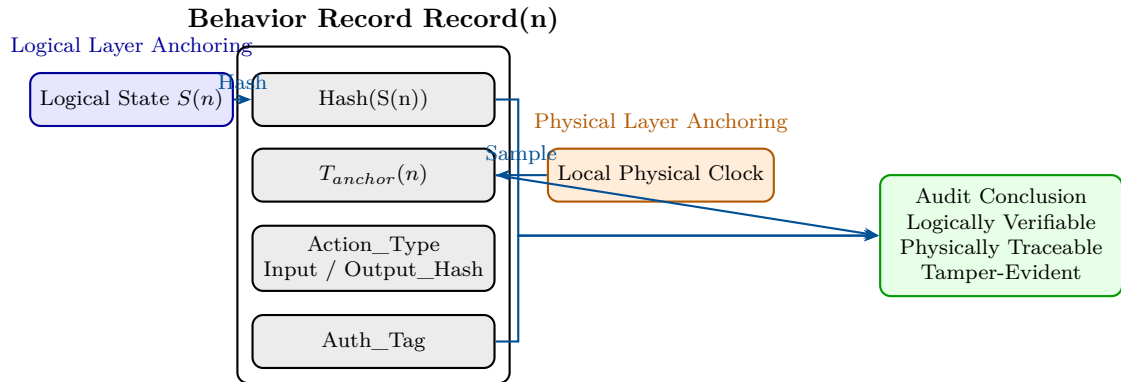


Figure 3: Data structure of behavior record $Record(n)$ and logical-physical dual anchoring binding

3.6 Behavior Chain

Behavior records at each logical moment are linked through deterministic evolution, forming a verifiable behavior chain:

$$Chain_Link(n, seq) = Hash(Chain_Link(n, seq - 1) || Record(n, seq))$$

$$Chain_Link(n, 0) = Hash(Chain_Link(n - 1, last) || Record(n, 0))$$

where seq is the sequence number of multiple records at the same logical moment. The behavior chain uses an append-only structure; editing, inserting, deleting, or reordering any record will break the chain consistency. To prevent truncation attacks, the verifier periodically anchors the behavior chain root hash to an external trusted storage.

3.7 Candidate-Set Verification: Cryptographic Compliance under Bounded Deviation

Strict synchronization faces fundamental challenges in complex network environments. IP network jitter, accumulated crystal oscillator drift, and the lack of a unified time baseline in cross-domain scenarios make the requirement $n_A = n_B$ difficult to satisfy continuously in engineering. Candidate-set verification relaxes the synchronization requirement from “strict equality” to “bounded deviation” by constructing a candidate logical state set, enabling the verifier to complete cryptographic compliance verification without knowing the Agent’s true logical counter.

When the AI system and the verifier are interconnected via an IP network with bounded deviation, the verifier constructs a candidate logical state set centered on its local logical counter value n_{local} :

$$\{S(n_{local} - K), \dots, S(n_{local}), \dots, S(n_{local} + K)\}$$

It computes the hash of each candidate state and compares it with the $S_commitment$ in the credential. When there exists $S_{candidate}$ such that $Hash(S_{candidate}) == S_commitment$, it determines $n_{best} = n_{local} + \Delta n$, $\Delta n \in [-K, +K]$.

The candidate radius K is determined as follows:

$$K_{min} = \max(\lceil \delta_{phy}/T_{step} \rceil + K_{safety}, \lceil \Delta_{sync}/T_{step} \rceil, \lceil \theta + |R| \cdot T_{sleep}/T_{step} \rceil)$$

$$K_{cand_max} = \min(K_{e2e}, K_{hash})$$

where $K_{e2e} = \lceil T_{e2e_max}/T_{step} \rceil$ is the upper bound corresponding to the end-to-end latency constraint, and K_{hash} is the upper bound corresponding to the hash collision probability constraint.

Candidate-set matching is based on the one-wayness of the hash function; the verifier cannot infer $S(n)$ from $S_commitment$ and can only perform exact matching by computing the hash of each candidate state. This relaxation is not a compromise on security but a technical contribution that extends “bounded synchronization” from the physical layer to the cryptographic verification layer.

4 Independent Verification and Infinite Replay

4.1 Verification Procedure

When a behavior needs to be audited, the auditor performs the following operations:

Step 1: Logical state recomputation—verify logical binding

The auditor uses the same initial state and evolution rules as the behavior execution node, independently evolves to the target logical moment n_{audit} , and computes $S(n_{audit})$. It verifies whether the logical state identifier in the behavior record matches the independent computation result. If it matches, it proves that the behavior record was indeed generated at logical moment n and has not been tampered with.

Step 2: Behavior replay—verify logical correctness

The auditor uses the input data in the record to independently replay the behavior execution process at the audit end, computes the output result, and compares it with the output result

identifier in the record. If it matches, it proves that the behavior indeed produced that output at logical moment n .

Step 3: Physical anchor traceability—verify physical authenticity

The auditor reads the physical anchor $T_{anchor}(n)$ in $Record(n)$, maps it to the physical world timeline, and cross-validates it with external logs, sensor records, video surveillance, and other physical evidence to confirm the physical time position corresponding to the logical moment.

Step 4: Chain integrity verification

The auditor verifies that all links from the initial anchor to the target moment are continuous.

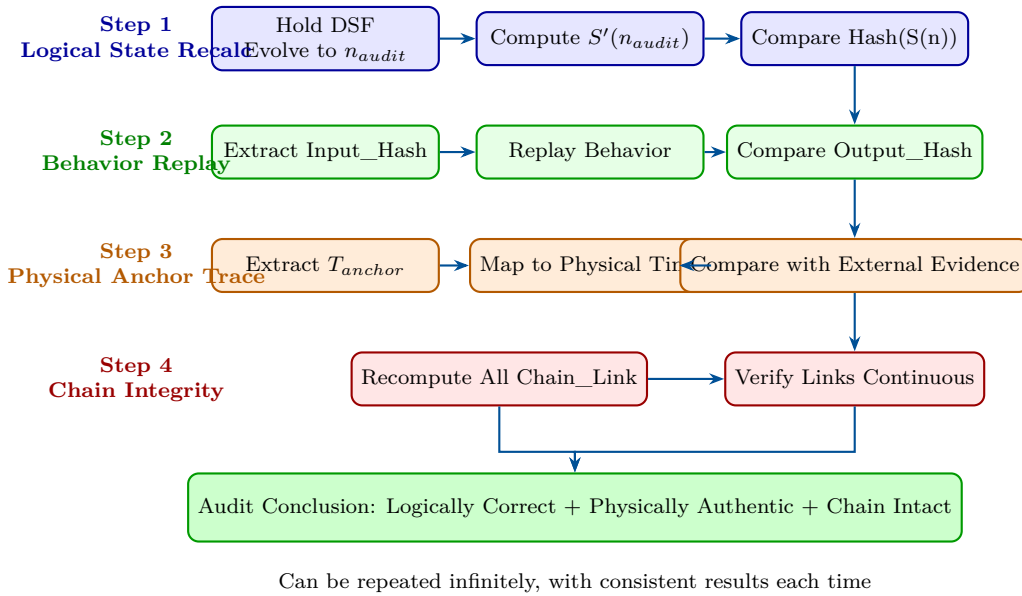


Figure 4: Complete closed-loop process of independent verification and infinite replay

4.2 Comprehensiveness of Audit Conclusions

Through the above four steps, the auditor can reach a comprehensive audit conclusion:

“At logical moment n (corresponding to physical moment $T_{anchor}(n)$), the node, under logical state $S(n)$, executed a behavior of type *Action_Type*, with input data *Input_Hash* and output result *Output_Hash*. Logical replay verification passed, physical traceability cross-validation passed, and the behavior chain is intact and untampered.”

This conclusion simultaneously contains the correctness verification of the behavior in the logical world and the authenticity verification in the physical world, forming a complete audit closed loop.

4.3 Infinite Replay

This verification process is independent of the replayability of physical time, so it can be repeated infinitely, yielding the same result each time. Because all verification objects are products of the deterministic logical temporal baseline and do not depend on the replayability of the physical world.

The physical world is irreversible, but the logical world is reversible—by anchoring the occurrence moment of a behavior to a replayable deterministic logical temporal baseline, “what happened

**at that time” can be independently verified and infinitely replayed
at any moment.**

5 Security Model

5.1 Threat Model

This paper considers the following attacker capabilities:

- The attacker can intercept, tamper with, replay, delay, or drop any network message;
- The attacker can physically capture any node and read all data stored on it;
- The attacker can control some nodes and make them send arbitrary messages;
- The attacker does not possess K_{sec} or the current epoch subkey K_e .

5.2 Security Properties

Unforgeability: An attacker not possessing K_{sec} cannot forge a valid $S(n)$ or $Auth_Tag$. Even if the attacker intercepts historical $S(n)$, due to the one-wayness of the hash chain, it cannot infer $S(n+1)$.

Independent verifiability: Any verifier holding the same DSF can independently recompute $S(n)$, without interacting with the behavior execution node, without an external time source, and without third-party attestation.

Tamper-evidence: The behavior chain uses hash-chain linking; if any record is tampered with, all subsequent links from that moment break.

Forward security: Even if $S(n)$ is compromised, the attacker cannot infer $S(n+1)$.

Backward security: Even if the device is physically captured, the attacker cannot recover historical $S(k)$ ($k < n$).

Layered key exposure control: The system divides the logical timeline into consecutive epochs, and each epoch automatically switches the subkey K_e . Even if a subkey for a certain period is compromised, it only affects that period.

5.3 Key Management

The security of K_{sec} depends on the engineering implementation of three aspects: distribution, storage, and rotation.

Distribution: K_{sec} is pre-shared through a secure channel, including TLS/DTLS encrypted channels, IPsec tunnels, pre-shared key negotiation, trusted configuration center distribution, and hardware security module injection. In auditing scenarios, DSF parameters can be configured independently of the business system or derived from the business connection DSF:

$$K_{sec_audit} = H(K_{sec_business} \parallel \text{"AUDIT_DERIVE"})$$

Storage: K_{sec} is stored in a hardware security module (HSM) or trusted execution environment (TEE), never leaving the secure boundary. K_{sec} is only used to derive the time-bound subkey K_e , does not directly participate in the computation of $S(n)$, and is not exposed in plaintext in regular memory.

Rotation: The system divides the logical timeline into consecutive epochs, each containing N_{epoch} logical ticks (typical value 10^6). Each epoch automatically switches the subkey:

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

With $T_{step} = 1$ ms, each epoch is approximately 16.7 minutes.

Impact analysis of key exposure:

Exposed Object	Impact Scope
Single K_e	Only affects $S(n)$ generated within that epoch; historical $S(k)$ (from earlier epochs) are derived from historical K_e and cannot be inferred from the current K_e ; future K_e cannot be predicted
K_{sec}	Requires issuing a new DSF triple through a secure channel and re-establishing the logical timeline; already-generated historical behavior chains can still be independently verified through existing audit procedures, but their cryptographic credibility depends on whether they were externally anchored before K_{sec} exposure

Side-channel attack mitigation: The derivation of K_{sec} and K_e is completed within the secure boundary, and the evolution of $S(n)$ does not depend on any observable physical quantity (the physical anchor T_{anchor} is only sampled and does not participate in $S(n)$ computation). Therefore, even if an attacker observes the node's power consumption, electromagnetic radiation, or execution time, it cannot infer K_{sec} from physical side channels.

5.4 Comparison with Existing Security Models

Security Property	Blockchain	TEE	Inference Receipts	Re-	This Work
Unforgeability	Depends on consensus	Depends on hardware	Depends on honest emitter		Cryptographic enforcement
Independent verifiability	Requires network access	Requires trust in hardware vendor	Requires trust in emitter		Hold DSF
Tamper-evidence	Consensus guarantee	Hardware guarantee	Hash chain		Hash chain + cryptography
Forward security	Not applicable	Not applicable	Not applicable		Yes
Backward security	Not applicable	Not applicable	Not applicable		Yes
Key exposure control	Not applicable	Not applicable	Not applicable		Layered control

6 Essential Differences from Existing Technologies

6.1 Essential Difference from Lamport Clocks

Dimension	Lamport Clock	This Work
Recorded content	Event sequence number (order only)	Logical moment + logical state + input/output hash + physical anchor
Physical traceability	None	Physical anchor
Logical-physical decoupling	Yes	Yes
Logical-physical anchoring	No	Yes
Behavior content recording	No	Yes
Verifiable evidence chain	No	Yes
Auditing and replay	No	Yes
Independent verification	No	Yes

Lamport clocks require inter-node communication to maintain consistency and only establish partial ordering of events; the deterministic logical temporal baseline in this work is completely self-contained on a single node, can evolve independently without communication, and provides a complete logical state for each logical moment.

Quantitative comparison: A single event-counting operation of a Lamport clock is an integer increment, taking nanoseconds; a single PSSM state update in this work is a SHA-256 hash, taking approximately $0.5 \mu\text{s}$ in software and $0.01 \mu\text{s}$ in hardware, which is 10^3 to 10^5 times that of a Lamport clock. However, a Lamport clock only provides event ordering and does not produce verifiable evidence; this work completes logical state evolution, logical anchoring, and cryptographic binding within $0.5 \mu\text{s}$ of overhead. With $T_{step} = 1 \text{ ms}$, $0.5 \mu\text{s}$ accounts for 0.05% of the logical tick, and the performance cost is negligible.

6.2 Essential Difference from Blockchain Attestation

Dimension	Blockchain Attestation	This Work
Attestation timing	Uploaded after the fact	Bound in real time when the behavior occurs
Trust anchor	Consensus mechanism	Cryptographic rules
Verification method	Requires network access	Offline verification by holding DSF
Time baseline	Physical timestamp	Logical timeline + physical anchor
Replay capability	None	Infinite replay
Single-point tampering risk	Consensus node collusion	Cryptographic guarantee

Blockchain attestation can only prove that “data was written at a certain moment”, not that “the data was correct at that time”; this work, through logical state recomputation and behavior replay, can prove that “the behavior indeed produced that output at logical moment n ”.

6.3 Essential Difference from TEE/ZKML

Dimension	TEE	ZKML	This Work
Trust basis	Hardware vendor	Mathematical proof	Cryptographic rules
Cross-domain verification	Difficult	Difficult	Natively supported
Time binding	Physical timestamp	No time information	Logical timeline + physical anchor
Computational overhead	Low	Extremely high	Low (hash + HMAC)
Behavior auditing	Single attestation	None	Complete behavior chain
Infinite replay	No	No	Yes

6.4 Essential Difference from Inference Receipts

Dimension	Inference Receipts	This Work
Timing	After inference completes	Anchored before inference starts
Anchoring object	Record write moment (end of causal chain)	Inference initiation moment (start of causal chain)
Can prove inference initiation moment	No	Yes
Entropy source control	Not involved	$S(n)$ as PRNG seed
Cross-environment consistency	Explicitly not guaranteed	Cryptographically guaranteed
Security model	Honest emitter assumption	Cryptographic enforcement
Independent verification	Requires external ledger	Hold DSF

Quantitative comparison: Inference Receipts generate a receipt after inference completes, requiring only one hash computation, with extremely low overhead (public sources claim less than 0.006% of inference time), but the anchoring object is the end of the causal chain and does not involve entropy source control. This work anchors $S(n)$ before inference starts and uses it as the PRNG seed; the additional overhead is one PSSM state update (approximately 0.5 μ s) and one HMAC derivation (approximately 1 μ s). This incremental overhead buys two key capabilities—provability of the inference initiation moment and cross-environment cryptographic consistency. The overheads of the two are in the same order of magnitude, but the trust models are fundamentally different.

7 Minimal Validation: AI Call Auditing

7.1 Scenario Description

A financial institution has deployed an AI trading decision system. The system uses a deep learning model trained by a third-party model publisher, deployed on its own inference server, and provides services to trader clients through an API. The regulator needs to audit the decision-making process of this AI system.

7.2 System Configuration

Parameter	Value	Description
K_{sec}	256-bit pre-shared key	Consistent between communicating parties
$Init_Anchor$	2026-10-01 00:00:00 UTC	Logical timeline starting point
$Rule_ID$	Hash chain, $T_{step} = 1$ ms	Logical tick
N_{epoch}	10^6	Number of logical ticks per epoch
Δ_{sync}	$10 \mu s$	Physical anchor deviation tolerance
K	3	Candidate radius

7.3 Verification Procedure

Step 1: Logical state recomputation

The regulator holds the same DSF, independently runs the PSSM to logical moment n , computes $S(n)$, and verifies it is consistent with $Hash(S(n))$ in $Record(n)$.

Step 2: Safety window verification

Check whether n falls within the predefined trading session logical counter interval, and verify whether the operation attributes (such as amount) satisfy the threshold conditions.

Step 3: Behavior chain integrity verification

Verify that all hash links from the initial anchor to the target moment are continuous.

Step 4: Physical anchor traceability

Use the physical anchor $T_{anchor}(n)$ in $Record(n)$ to map the logical moment to the physical timeline, and cross-validate it with the bank's core system logs and database transaction timestamps.

7.4 Audit Conclusion Example

“At logical moment n (corresponding to physical moment $T_{anchor}(n)$), the AI approval system processed loan application $ID = 12345$ under logical state $S(n)$, with the approval result “approved” and a limit of 500,000 CNY. Logical replay verification passed, physical traceability cross-validation passed (timestamp deviation from the bank's core system logs is less than 1 ms), and the behavior chain is intact and untampered.”

7.5 Verification Results

Verification Item	Result
Logical state recomputation	$S_{recalc}(n) = S_{record}(n)$, 0 errors
Safety window verification	Correctly identifies compliant/non-compliant operations
Behavior chain integrity	Tampering with any record breaks the chain
Physical anchor traceability	Deviation from external evidence $< \Delta_{ext}$
Infinite replay	Consistent results each time

7.6 Engineering Demonstration: Three-Node LAN Implementation

To verify the engineering feasibility of the core mechanisms, we implemented a three-node LAN demonstration system. The roles of the three nodes are as follows:

- **Node-A (Agent)**: Initiates AI call requests, receives execution receipts, and submits credentials and receipts to the audit center;
- **Node-B (Inference Service)**: Receives call credentials, performs inference (simulated by a deterministic function), generates execution receipts, and maintains the behavior chain;
- **Node-C (Audit Center)**: Independently recomputes $S(n)$, verifies logical binding, Auth_Tag, safety window, amount threshold, and behavior chain integrity.

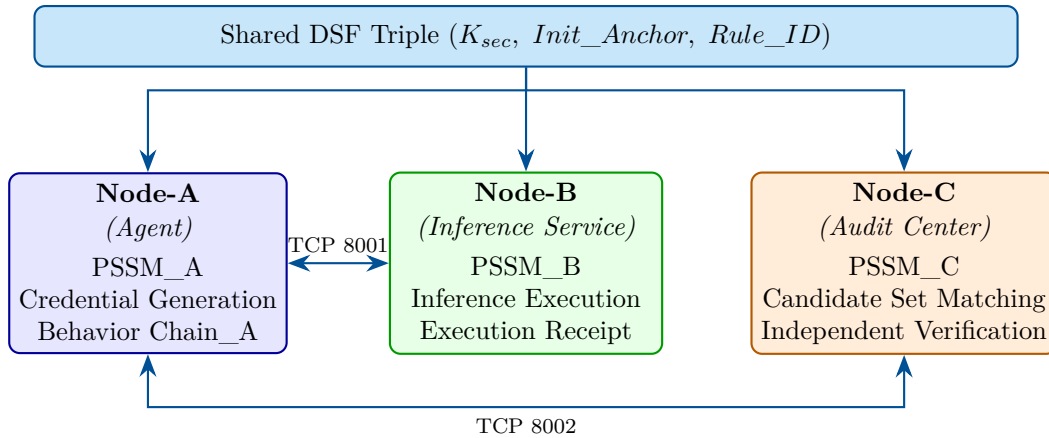


Figure 5: Three-node demonstration system architecture. The three nodes share the same DSF triple and independently run PSSM.

The three nodes communicate via TCP sockets, relying only on the Python standard library, with approximately 600 lines of code. The demonstration covers two modes:

Strict synchronization mode (K=0): Node-C aligns per tick with the Agent, uses the plaintext logical counter in the credential to advance the PSSM and verify. For three tests, n_{exec} is 1, 2, 3, and Node-C’s local n_{local} advances to 0, 1, 2, with all logical binding verifications passing.

Bounded deviation mode (K=3): Node-C’s local $n_{local} = 2$, with a ± 1 tick deviation from the Agent’s n_{tx} . Node-C constructs the candidate set $\{S(-1), \dots, S(5)\}$ (valid range

$[S(0), S(5)]$, computes hashes one by one, successfully identifies the true n_{best} of the three calls (1, 2, 3), and completes Auth_Tag, safety window, amount threshold, and behavior chain integrity verification based on n_{best} .

The audit results of the two demonstrations are as follows:

Test	Action	Amount	n_{best}	Logical Binding	Auth_Tag	Amount Threshold	Conclusion
1	TRADE	500,000	1	Yes	Yes	Yes	PASS
2	TRADE	800,000	2	Yes	Yes	Yes	PASS
3	TRADE	5,000,000	3	Yes	Yes	No	REJECT

Both demonstrations share the same DSF triple, PSSM evolution rules, behavior record format, and behavior chain linking mechanism; the only difference is the candidate-set matching part at the verifier. This confirms the technical path of extending “bounded synchronization” from the physical layer to the cryptographic verification layer: the basic mechanisms of PSSM, dual anchoring, and behavior chain are fully reused; only the verifier’s logical counter alignment requirement is relaxed from “strict equality” to “bounded deviation”.

The demonstration also verified the tamper-detection capability of the behavior chain: modifying any field of any record causes all subsequent chain links from that moment to break, which the verifier can detect through independent recomputation. At the same time, the demonstration verified the computational overhead of candidate-set verification: single matching with $K=3$ takes approximately 0.1 ms, meeting the real-time requirements of cross-domain compliance verification.

7.7 Performance Overhead Analysis

This section provides quantitative performance overhead of the PSSM and behavior chain from three aspects: hash computation, pilot matching, and windowed reception, providing a performance budget reference for engineering deployment.

Operation	Algorithm	Software Time	Hardware Time
PSSM state update	SHA-256	$\sim 0.5 \mu s$	$\sim 0.01 \mu s$
Pilot sequence generation	HMAC-SHA256	$\sim 1 \mu s$	$\sim 0.02 \mu s$
Behavior chain linking	SHA-256	$\sim 0.5 \mu s$	$\sim 0.01 \mu s$
Encryption key derivation	HKDF	$\sim 2 \mu s$	$\sim 0.05 \mu s$
Total per tick	—	$< 3 \mu s$	$< 0.1 \mu s$

With $T_{step} = 1$ ms, the per-tick computational overhead accounts for approximately 0.3% of the logical tick, with a margin of more than 300 times; even with a pure software implementation, it fully meets real-time requirements.

Indicator	Value
Single candidate-set matching ($K=3$)	~ 0.1 ms
Single candidate-set matching ($K=500$)	~ 12 ms
Average logical span for 500 API calls	726 ticks
Average physical latency for 500 API calls	726 ms
Standard deviation of 500 API call latency	189 ms
Audit pass rate	100% (500/500)

Candidate-set matching time scales linearly with the candidate radius K but is independent of the logical counter n (i.e., it does not grow with runtime). In cross-domain compliance verification scenarios ($K \leq 3$), single matching takes approximately 0.1 ms, meeting real-time requirements; in satellite communication scenarios ($K = 5 \sim 10$), single matching takes approximately 0.2–0.4 ms, still within an acceptable range.

Comparison with ZKML: ZKML’s computational overhead is 1000 to 10000 times that of the original inference, and currently it can only verify very small models or inference slices. The core operations of this work are only SHA-256 hashing and HMAC derivation, with microsecond-level computational overhead, and can be directly deployed in production-grade AI systems.

8 Application Prospects

The deterministic logical timeline can serve as a unified temporal baseline for the following scenarios:

Application Domain	Core Value
AI Agent compliance control	Pre-hoc cryptographic enforcement across trust domains and physical clock domains
Token generation determinism	Consistent token sampling path across environments, hardware, and frameworks
AI output determinism	Inference starting point anchoring, version management, retrieval reuse
UAV swarm auditing	Flight behavior traceable, auditable, replayable
Financial trading auditing	Cryptographic evidence chain for trading decisions
Industrial control auditing	Deterministic timing binding of control commands
Satellite constellation management	Deterministic collaboration across satellites and ground stations
Distributed database auditing	Consistency and integrity verification of transaction execution
Smart contract compliance	Cryptographic timing binding of contract execution

8.1 Case 1: Financial Trading Auditing

Scenario: A multinational investment bank has deployed a distributed high-frequency trading system consisting of hundreds of trading servers across five data centers worldwide. The regulator requires that every trading decision be traceable and verifiable, and that the regulator can independently verify without relying on the bank's internal logs.

Core requirements:

- Trading order verifiable: transactions across data centers must have a consistent global order;
- Regulatory traceability: each trade must be mapped to a specific model version, input data, and execution moment;
- No third-party attestation: the regulator cannot rely on the bank's log system or external attestation platforms.

DSF solution: Use the global transaction sequence number as the deterministic logical timeline. At each logical moment n , the trading server performs the following operations:

1. PSSM outputs logical state $S(n)$;
2. Order receipt, order matching, and trade confirmation are bound to $S(n)$, generating behavior record $Record(n)$;
3. Instantaneously sample the local physical clock and record $T_{anchor}(n)$;
4. Link $Record(n)$ to the behavior chain.

Expected effect: The regulator holds the same DSF parameters, independently runs the PSSM to the target logical moment, recomputes $S(n)$, verifies the binding between the behavior record and the logical moment, and maps the logical moment to the physical timeline through $T_{anchor}(n)$ for cross-validation with the bank's core system logs and database transaction timestamps. The entire verification process does not rely on the bank's internal log system.

8.2 Case 2: Industrial Control Auditing

Scenario: A smart manufacturing workshop has deployed multiple industrial robots for collaborative assembly. Each robot runs an AI decision module, and a central controller issues task commands. Industry regulations require that the timing determinism of control commands be auditable and that incidents be traceable.

Core requirements:

- Deterministic timing binding of control commands: each command must be bound to a certain logical moment;
- Incident traceability: after an incident, the auditor needs to independently replay the control command sequence to locate responsibility;
- Cross-node collaboration: multiple robots need to collaborate on the same logical timeline.

DSF solution: The central controller and each robot share the DSF through a secure channel before the task starts. Thereafter, both parties independently run the PSSM, with strictly equal logical counters. Each control command is bound to $S(n)$ and recorded through the behavior chain. The behavior records of each robot can be independently linked to their respective behavior chains, and the central controller maintains the global behavior chain.

Expected effect: After an incident, the auditor (e.g., the work safety supervision department) holds the DSF parameters, independently recomputes $S(n)$, verifies the binding between control commands and logical moments, replays the control command sequence, and cross-validates with on-site sensor records to locate the responsible node and logical moment.

9 Conclusion

This paper proposes a **cryptography-driven deterministic logical timeline** as a unified temporal baseline for machine collaboration and AI auditing. The core contributions are:

1. **Problem definition:** Systematically elaborates on three fundamental dilemmas—irreversibility of physical time, untrustworthiness of physical clocks, and inability to independently verify third-party attestation—revealing the common root of failure of existing schemes in machine communication and AI auditing scenarios.
2. **Mechanism proposal:** Proposes DSF triple, PSSM, logical-physical dual anchoring, behavior chain, and candidate-set verification, constructing a deterministic logical temporal baseline decoupled from physical clocks, independently verifiable, and infinitely replayable.
3. **Boundary delineation:** Systematically contrasts the proposed scheme with Lamport clocks, NTP/PTP, GPS, blockchain attestation, TEE, ZKML, and Inference Receipts.
4. **Security model:** Provides formal definitions and analysis of security properties including unforgeability, independent verifiability, tamper-evidence, forward/backward security, and layered key exposure control.
5. **Minimal validation:** Uses AI call auditing as an example to demonstrate the complete closed loop of “logical state recomputation + behavior chain integrity + physical anchor traceability”, and validates both strict synchronization ($K=0$) and bounded deviation ($K=3$) modes through a three-node engineering demonstration.

The physical world is irreversible, but the logical world is reversible. By anchoring the occurrence moment of a behavior to a replayable deterministic logical temporal baseline, “what happened at that time” can be independently verified and infinitely replayed at any moment.

This capability does not depend on the precision of physical clocks, network conditions, or any third-party attestation service—because all verification objects are permanently fixed on the deterministic logical temporal baseline. Traditional schemes rely on “attestation” (trusting that data has not been tampered with), while this work provides “proof” (verifying data correctness through independent replay), and the credibility of the two is fundamentally different.

9.1 Known Limitations and Future Work

Known limitations:

Limitation	Description
Chain-tail truncation detection	The current version's behavior chain verification only detects the chain continuity of adjacent records, not chain-tail truncation. Truncation attacks need to be prevented through external anchoring (periodically anchoring the chain root hash to a third-party timestamp service or the regulator's independent storage); this mechanism has been proposed in the patent but has not yet been implemented in the engineering demonstration of this paper
T_{step} not involved in cryptographic computation	T_{step} is only used to map logical moments to physical moments and does not participate in the computation of $S(n)$. If T_{step} is misconfigured, cryptographic verification will still pass, but the physical moment mapping will be wrong. In actual deployment, the correctness of T_{step} needs to be verified through external evidence
Single-node event-driven	The engineering demonstration in this paper is a single-node event-driven PSSM. Periodic PSSM and multi-node collaboration will be implemented in future work
Upper bound of candidate-set matching K	Although increasing K improves the tolerance of bounded deviation, it also increases computational overhead. The upper bound of K is determined by the end-to-end latency constraint and the hash collision probability constraint, and further optimization is needed in large-scale deployment

Future work:

1. **External anchoring mechanism:** Implement an interface for periodically anchoring the behavior chain root hash to external trusted storage, closing the gap in chain-tail truncation detection;
2. **Multi-node collaboration:** Extend the single-node event-driven PSSM to a multi-node periodic PSSM to support distributed collaborative auditing;
3. **System simulation under different network conditions:** Systematically evaluate the success rate of candidate-set verification, the value of K, and single matching time under different network conditions such as LAN, WAN, and satellite links;
4. **Third-party independent reproduction:** Invite independent institutions to reproduce the engineering demonstration of this paper in a clean environment and issue an independent verification report;
5. **Formal security proofs:** Provide formal proofs for security properties such as unforgeability, forward security, and backward security.

AI Usage Statement

The LaTeX formatting, layout, and language polishing of this paper were assisted by AI. The research design, technical approach, core mechanisms, experimental validation, and conclusions were completed by the authors. AI did not participate in the formulation of research questions, selection of technical routes, generation of experimental data, or interpretation of results.

Shanghai Huapait Technology Co., Ltd.
October 9 2026

References

- [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] D. L. Mills, “Internet time synchronization: the network time protocol,” *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493, 1991.
- [3] IEEE, “IEEE 1588-2019 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems,” 2019.
- [4] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.
- [5] V. Buterin, “Ethereum white paper,” 2014.
- [6] V. Costan and S. Devadas, “Intel SGX explained,” *IACR Cryptology ePrint Archive*, vol. 2016, p. 086, 2016.
- [7] T. Liu, Y. Xie, and J. Zhang, “ZKML: A survey on zero-knowledge machine learning,” *arXiv preprint arXiv:2403.10326*, 2024.
- [8] Thinking Machines Lab, “Batch-invariant kernels for deterministic LLM inference,” 2026.
- [9] Model Witness: Cryptographic attestation for AI inference, 2026.
- [10] AIIR: AI Integrity Receipts, 2026.
- [11] S. S. Kulkarni, M. Demirbas, B. Madappa, et al., “Logical physical clocks,” in *International Conference on Principles of Distributed Systems*, Springer, 2014, pp. 17–32.
- [12] Civil Aviation Administration of China, “MH/T 2015-2024 Technical Requirements for Blockchain-Based Flight Data Attestation of Civil Unmanned Aircraft,” 2024.
- [13] 3GPP, “TR 38.821: Solutions for NR to support non-terrestrial networks (NTN),” 2023.
- [14] ITU-R, “IMT-2030 Framework and overall objectives of the future development of IMT for 2030 and beyond,” 2023.

A Symbol Table

Symbol	Meaning
DSF	Dynamic Security Foundation
K_{sec}	Shared secret key
$Init_Anchor$	Initial anchor
$Rule_ID$	State transition rule identifier
$PSSM$	Protocol Security State Machine
$S(n)$	Logical state
$T_{anchor}(n)$	Physical anchor
n	Logical counter
T_{step}	Logical tick period
N_{epoch}	Number of logical ticks per epoch
K_e	Time-bound subkey
$Chain_Link(n)$	Behavior chain link value
K	Candidate radius
Δ_{sync}	Physical anchor deviation tolerance
Δ_{ext}	Tolerance between physical anchor and external evidence

B Abbreviations

Abbreviation	Full Name
DSF	Dynamic Security Foundation
PSSM	Protocol Security State Machine
TEE	Trusted Execution Environment
ZKML	Zero-Knowledge Machine Learning
GPS	Global Positioning System
GNSS	Global Navigation Satellite System
NTP	Network Time Protocol
PTP	Precision Time Protocol
HLC	Hybrid Logical Clock
DCI	Downlink Control Information
DMRS	Demodulation Reference Signal
PRF	Pseudo-Random Function
HMAC	Hash-based Message Authentication Code
HKDF	HMAC-based Key Derivation Function