Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 10 October 2026

H. Jorgen
Kenosian

10 April 2026

The TLS TimeToken Secure Protocol (tttps://) draft-helmprotocol-tttps-02

Abstract

This document specifies the TLS TimeToken Secure Protocol (tttps://), a protocol extension that augments TLS 1.3 [RFC8446] with cryptographically verifiable temporal ordering.

Internet infrastructure assumes that channels are passive: noise is random and channel operators have no ordering preferences. This assumption is structurally violated when ordering has economic value -- NTP servers, BGP routing authorities, DNS resolvers, and transaction sequencers all have incentive to misrepresent ordering. This document formalises the problem as the Strategic Channel Controller Problem (SCCP), absent from classical information theory.

Temporal ordering attacks are structurally more acute for autonomous AI agents than for human participants: as agent reaction times converge toward symmetry, ordering advantage can no longer be earned through superior human latency. No existing protocol -- including O(n^2) BFT consensus, which tolerates but does not eliminate Byzantine nodes -- provides a cryptographic pre-ingestion defense for this case.

TTTPS introduces Proof-of-Time (PoT): a multi-source synthesised timestamp protected by the GRG integrity pipeline (Golomb-Rice -> Reed-Solomon -> Golay(23,12,7) -> HMAC), whose stage ordering is mathematically necessary (Theorems 1-3 of the companion paper [POT2026]). PoT achieves Byzantine temporal elimination at O(1) per record, independent of network size. An AdaptiveSwitch mechanism makes ordering manipulation economically self-defeating; the equilibrium threshold is derived in closed form and empirically calibrated from deployed data (Section 6.4).

Deployment on Base Sepolia produces 70,000+ verified records; 55% are generated by autonomous AI agents -- an unanticipated finding that confirms the structural severity of the ordering problem in agent economies.

This document has Experimental status. The GRG pipeline specification will be published upon conclusion of pending patent proceedings (Section 12).

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at https://www.ietf.org/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at https://www.ietf.org/shadow.html

This Internet-Draft will expire on 10 October 2026.

==========
TABLE OF CONTENTS
==========

DISCUSSION NOTE

This document is being discussed on the dispatch@ietf.org mailing list. The authors intend to request a Birds-of-a-Feather (BoF) session to explore working group formation for temporal ordering protocols. Comments and participation are welcome.

SECTION 1. INTRODUCTION

The Internet’s trust infrastructure rests on two pillars. Identity trust, solved by TLS/PKI, answers "who" sent a message. Temporal trust remains unsolved at the protocol level: "when" did a message originate, in a manner no single party can falsify?

This gap was irrelevant in 1948 when Shannon modelled channels as passive. It became critical as networks were built by economic agents: NTP servers can bias timestamps to manipulate financial settlement windows; BGP routing authorities can reroute traffic [RFC4272]; DNS resolvers exploit temporal priority. Autonomous AI agents executing hyperagent architectures generate ordering-sensitive transactions at machine speed with no human audit cycle.

These are not independent failures. They are instances of one structural property: in any network where message ordering has economic value, the ordering authority has incentive to misrepresent it. Shannon’s channel theory has no model for this.

1.1. Objectives

The objectives of TTTPS are as follows:

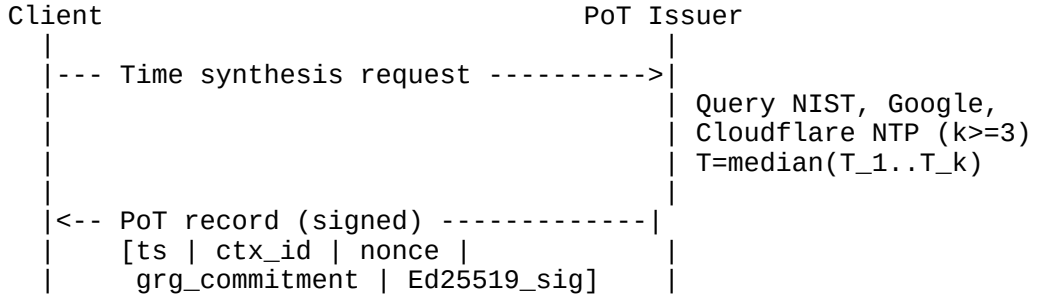
- Temporal origin authentication: prove "when" a message originated, complementing TLS's proof of "who".
- Byzantine time source elimination: transform detection probability from P(detect) < 1 (Shannon model) to P(detect) >= 1 - 2^{-61} via GRG context binding.
- Delay attack prevention: enforce that PoT submissions outside the tier tolerance window are rejected pre-ingestion, as defined for delay attacks in [RFC8915] Section 8.6.
- Economic eviction of dishonest nodes: via AdaptiveSwitch equilibrium threshold V*, below which ordering manipulation is self-defeating.
- Transport-layer agnosticism: operate over TLS 1.3 [RFC8446], QUIC [RFC9000], and HTTP/3 [RFC9114] without modification to those protocols.
- Backward compatibility: deployable alongside existing TLS 1.3 without requiring server-side changes.
- Experimental deployment: accumulate implementation experience prior to consideration for the Standards Track.
- Privacy-preserving temporal attestation: PoT binds to context without revealing transaction content or participant identity.

Primary use cases include MEV-resistant decentralised exchange (DEX) transaction ordering, AI agent-to-agent payment sequencing, IoT mission-critical command ordering, and financial settlement timestamping.

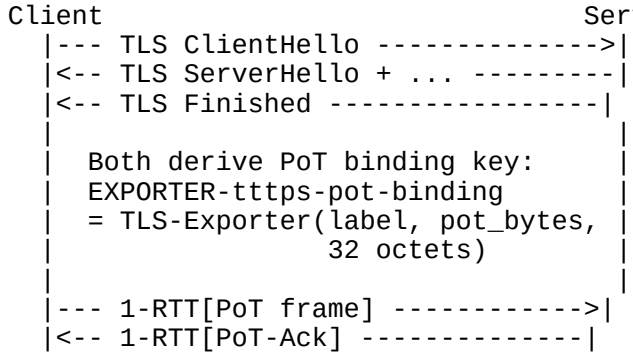
1.2. Protocol Overview

TTTPS operates in two phases:

Phase 1 -- PoT Generation:



Phase 2 -- TLS Binding (TLS Exporter, RFC 5705):



Byzantine nodes that submit manipulated ordering are identified with probability >= 1 - 2^{-61} and economically penalised via AdaptiveSwitch FULL mode.

TTTPS does NOT modify the TLS handshake. No new TLS Extension Type is required. This approach follows RFC 8915 Section 5.1.

1.3. Scope

This document specifies:

- The PoT data structure and wire format (Section 4)
- The GRG Integrity Pipeline abstract interface (Section 5)
- The AdaptiveSwitch Byzantine eviction mechanism (Section 6)
- The TTTPS transport binding (Section 7)

This document does NOT specify:

- Concrete implementation of GRG pipeline cryptographic operations (covered by pending patent; see Section 12)
- Specific NTP server selection policies
- Smart contract implementations for on-chain anchoring
- Pricing or fee schedules (implementation-defined; Section 8)
- Satellite time source integration (future work)

1.4. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are interpreted as described in BCP 14 [RFC2119] [RFC8174].

SCCP (Strategic Channel Controller Problem):
: A system satisfies SCCP if (i) a controller C has authority over message ordering; (ii) U(C) is strictly monotone in that ordering; (iii) no external party can verify original ordering without C's cooperation. Instances include NTP timestamp bias, BGP hijacking, DNS poisoning, and transaction sequencer MEV.

Proof-of-Time (PoT):
: A cryptographically authenticated record of a synthesised timestamp, bound to a context identifier via GRG context binding, and protected against replay and delay.

GRG Pipeline:
: A four-stage integrity pipeline: G_1 (Golomb-Rice encoding), R (Reed-Solomon erasure coding), G_2 (Golay(23,12,7) forward error correction), H (HMAC-SHA256 context binding). The stage ordering is mathematically necessary (Section 5.4). Implementation is proprietary; only the abstract interface and external properties are specified here.

AdaptiveSwitch:
: A state machine classifying nodes as TURBO (ordering-compliant, ~50 ms verification, 20% fee discount) or FULL (potentially Byzantine, ~127 ms, exponential backoff).

Byzantine Time Attack:
: An adversarial action in which a network participant reports a fabricated or manipulated timestamp to gain ordering advantage.

V* (Equilibrium threshold):
: V* = c_0 + lambda * Delta_tau. For MEV opportunity value V < V*, ordering manipulation is eliminated in the unique symmetric Nash equilibrium. Empirically calibrated from

151,423 Timeboost auctions: V* in [$8.67, $87.13].

FILO+GRG:
The processing discipline in which PoT submissions are subject to two sequential gates (HMAC context gate, then AdaptiveSwitch recency gate) before entering the processing queue; within the queue, the most recently generated qualifying PoT is processed first.

PoT Issuer:
An entity authorised to generate and sign PoT records. Analogous in function to a Certificate Authority, but attesting time rather than identity.

Tier:
An ordered set of time resolution levels (T0_epoch through T3_micro) controlling the tier tolerance window for PoT submission recency. See Section 8.

SECTION 2. REQUIREMENTS LANGUAGE

The key words "MUST", "MUST NOT", etc. in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals.

SECTION 3. PROBLEM STATEMENT

3.1. The Shannon Gap: SCCP

Shannon (1948) modelled a channel as Y = X + N_rand, where noise N is random and the channel operator is passive. All subsequent coding theory, information theory, and cryptographic channel models assume NOT-SCCP.

This assumption is structurally violated by modern internet infrastructure. Table 1 maps documented attack classes to SCCP Definition 1.1.4.

Table 1: SCCP instances in deployed infrastructure.

Domain	SCCP mechanism	lambda
NTP server	Timestamp bias shifts settlement windows	Low
BGP router	Traffic rerouting disrupts ordering	Medium
DNS resolver	Forged response wins temporal race	Low
Tx sequencer	Reordering extracts MEV	$0.11--1.13/ms
AI agent coordinator	Agent ordering captures surplus	High (scaling)

Shannon’s noise model has no mechanism for strategic N. PoT changes this: Byzantine manipulation becomes cryptographically self-identifying and economically self-penalising.

3.2. Existing Mitigations and Their Limitations

Table 2: Comparison with existing temporal protocols.

	Roughtime	NTS/PTP	PoT
Timing	Retro.	Session	Pre-ingestion
Enforcement	Signal	Reject	Economic penalty
Cross-domain	No	No	Yes
SCCP	No	No	Yes
Complexity	O(log n)	O(1)	O(1) per record
BFT overhead	N/A	N/A	O(1) vs O(n^2)

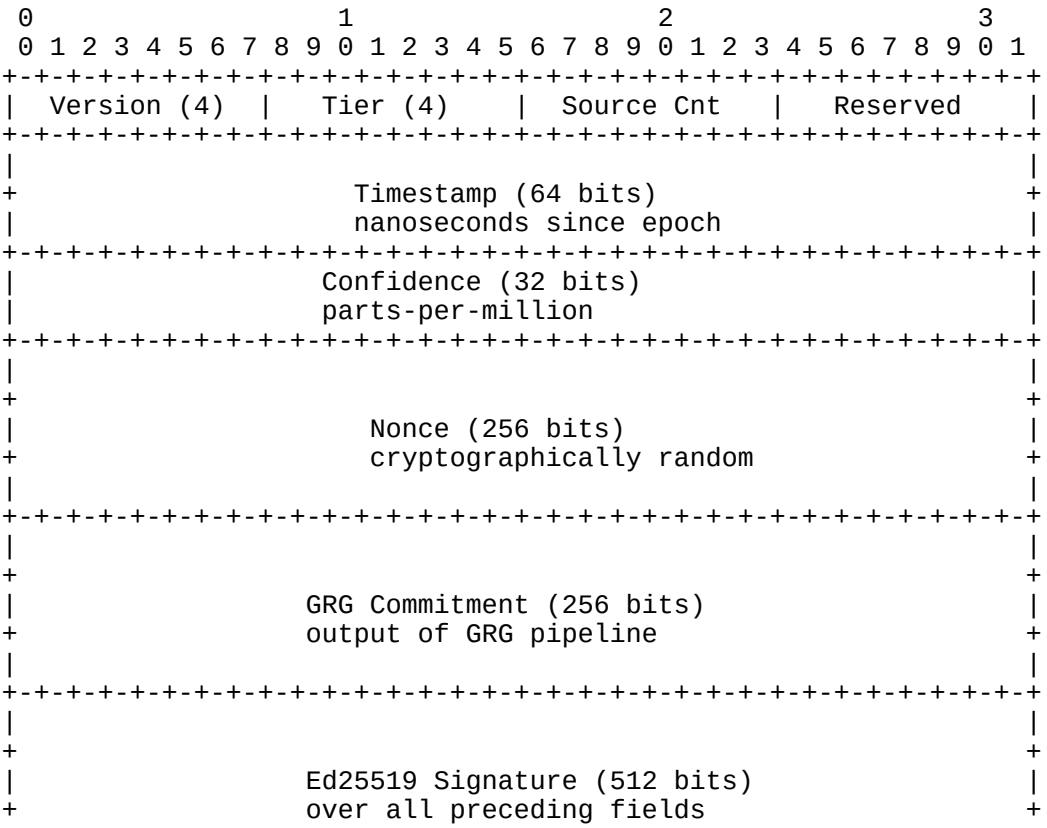
All existing BFT protocols (PBFT [RFC], Tendermint, HotStuff) TOLERATE f < n/3 Byzantine nodes with O(n^2) message complexity. PoT ELIMINATES Byzantine ordering manipulation at O(1) per record: the manipulation is not outvoted but cryptographically identified and economically penalised. As network size grows from n to 10n, BFT overhead grows 100x; PoT overhead is unchanged.

The urgency of this gap has increased as AI systems acquire autonomous capability to identify and exploit protocol-layer vulnerabilities without human direction [GLASSWING]. The same agentic capabilities that enable autonomous security research equally enable autonomous temporal attack on ordering-sensitive networks. TTTPS provides a pre-ingestion cryptographic defense that is independent of network size and agent count.

SECTION 4. PROOF-OF-TIME STRUCTURE

4.1. PoT Wire Format

A PoT record is encoded as a fixed-structure binary sequence. All multi-byte integer fields are in network byte order (big-endian).



```
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Total: 3 + 8 + 4 + 32 + 32 + 64 = 143 bytes.
(Version[1B] + SourceCnt[1B] + Reserved[1B] = 3 bytes
for the header row; all remaining fields as shown.)

4.2. Field Definitions

Version (4 bits):
Protocol version. This document defines version 1 (0x1).
Implementations MUST reject PoT records with unknown versions.

Tier (4 bits):
Identifies the time resolution level (Section 8).
Values: 0x0 = T0_epoch, 0x1 = T1_block, 0x2 = T2_slot,
0x3 = T3_micro, 0x4 = T-s1 (Deep-Space, Earth-Moon RTT).
Values 0x5-0xF are reserved for future deep-space tiers.

Source Count (8 bits):
Number of independent NTP sources consulted.
MUST be >= 3. Implementations SHOULD use >= 4.

Reserved (8 bits):
MUST be set to 0x00 by senders.
Receivers MUST ignore this field.

Timestamp (64 bits):
Synthesised timestamp: T_synth = median(T_1, ..., T_k),
k >= 3 sources from independent domains. Nanoseconds
since Unix epoch. Synthesis MUST use at least three
independent sources from distinct administrative domains
(e.g., NIST, Google, Cloudflare).

Confidence (32 bits):
Synthesis quality metric in parts-per-million. Computed
from inter-source agreement. Values above 1,000,000 ppm
MUST NOT be issued.

Nonce (256 bits):
Cryptographically random value. MUST be generated with
a cryptographically secure random number generator.
Provides replay prevention in conjunction with Section 9.2.

GRG Commitment (256 bits):
Output of the GRG Integrity Pipeline (Section 5) applied
to the preceding fields. The commitment cryptographically
binds the PoT payload to its context (chain_id, pool_address).

Ed25519 Signature (512 bits):
Signature over all preceding fields using the PoT Issuer's
Ed25519 private key [Bernstein2012], following EUF-CMA
security. The signature seals the GRG Commitment (double
seal property, Section 5.2).

Issuer Integrity Property: The Issuer cannot forge a
timestamp without detection. A forged timestamp T' != T
produces a different payload P', which produces a different
GRG_Commitment' != GRG_Commitment. The Ed25519 signature
over (P || GRG_Commitment) then fails verification against
the published Issuer public key. This is a mathematical
consequence of EUF-CMA security, not a procedural control.

The remaining trust assumption is that the Issuer's private
key is not compromised. This is the same trust model as
PKI (Certificate Authority), applied to time rather than

identity. Issuer misbehaviour is auditable via the on-chain commitment (Section 4.4).

4.3. Generation Algorithm

1. Query k >= 3 NTP sources from independent domains.
2. Compute T_synth = median(T_1, ..., T_k).
3. If max|T_i - T_synth| > stratum_tolerance: ABORT.
4. Generate 256-bit cryptographically random Nonce.
5. Assemble payload P = [Version | Tier | Source_Cnt | Reserved | Timestamp | Confidence | Nonce].
6. Compute GRG_Commitment = GRG(P, ctx_id) (Section 5).
7. Compute Sig = Ed25519.Sign(sk, P || GRG_Commitment).
8. Output PoT = P || GRG_Commitment || Sig.

4.4. On-Chain Commitment

TTTPS is a TLS-layer protocol. The on-chain component serves as a tamper-evident audit log -- not as the application target or consensus mechanism. Blockchain is the verification substrate; it records PoT commitments for independent audit.

The on-chain anchor is the keccak256 hash of the PoT record:

```
on_chain_hash = keccak256(ABI_encode(
    timestamp, grg_commitment, ctx_id, nonce))
```

This provides an immutable public record independently of the TTTPS protocol layer. Verifiers MAY cross-check the on-chain record to audit Issuer behaviour over time. Deployments that do not require on-chain audit MAY omit this step; the core protocol (Sections 4.1-4.5) functions without it.

4.5. Verification

Implementations MUST verify PoT records in the following order:

1. Version check: reject unknown versions.

1a. TLS binding verification (normative, if PoT frame includes binding_key per Section 7.1):

```
expected_key = TLS-Exporter(
                  "EXPORTER-tttps-pot-binding",
                   pot_record_without_sig,
                   32)
```

If expected_key != binding_key_in_frame: REJECT. Rationale: prevents cross-session replay. A PoT generated in TLS session A is cryptographically bound to that session and cannot be replayed in session B.

1b. Roughtime chain integrity (normative): The PoT Issuer MUST derive its synthesised timestamp from a Roughtime chain [ROUGHTIME] of k >= 3 independent servers:

```
D_chain = SHA-256(k Roughtime attestations)
GRG_Commitment = GRG(P || D_chain, ctx_id)
```

Theorem 0 (Inflow-to-Proof): A forged timestamp T' != T produces P' != P, hence D_chain' != D_chain, hence GRG_Commitment' != GRG_Commitment. The Ed25519 signature over (P || GRG_Commitment) then fails verification. Issuer timestamp manipulation is therefore mathematically detectable, not merely procedurally controlled.

2. HMAC context gate (~6 microseconds):
   Recompute HMAC(k, shard_i) for all shards.
   If any HMAC fails: REJECT immediately.
   DO NOT proceed to Ed25519 verification.
   NOTE: HMAC-first yields 16x cost reduction on invalid input.

3. Ed25519 signature verification (~46 microseconds in Rust):
   Verify Sig over the full PoT record.

4. Recency check (AdaptiveSwitch gate, Section 6):
   If submission_time - timestamp > tier_tolerance: REJECT.
   Trigger FULL mode per Section 6.3.

5. Nonce freshness: reject duplicate nonces (Section 9.2).

NOTE: The HMAC-first verification order (step 2 before step 3) achieves 16x cost reduction on invalid submissions. Invalid context binding -- which includes delayed resubmissions of valid PoTs in a different execution context -- is detected at the HMAC layer without incurring Ed25519 overhead.

SECTION 5. GRG INTEGRITY PIPELINE

5.1. External Interface

The GRG pipeline accepts a payload P and a context identifier ctx_id, and produces a 256-bit commitment:

```
GRG_Commitment = GRG(P, ctx_id)
```

Implementations of the GRG interface MUST satisfy:

- Lossless round-trip: GRG_Inverse(GRG(P)) = P
- Erasure tolerance: any k of n shards reconstruct P (where k and n are implementation-defined, minimum k=4, n=6)
- Bit-error correction: up to t=3 bit errors per 23-bit block are corrected
- Context binding: GRG(P, ctx_id_A) != GRG(P, ctx_id_B) for ctx_id_A != ctx_id_B, with probability >= 1 - 2^{-61}

Full pipeline specification is in Appendix B. Reference implementation: [OPENTTT].

5.2. Context Binding

The HMAC context key is derived as:

```
k = keccak256(chain_id || pool_address)
```

NOTE: This key is publicly derivable by design.
Purpose: domain separation (context binding), NOT secrecy.
Security model:
  Confidentiality/Authenticity = Ed25519 private key (Issuer)
  Context binding   = HMAC key (public, deterministic)

Attack prevented: A PoT shard from pool A cannot be replayed into pool B even if an attacker knows both HMAC keys, because the Ed25519 signature over grg_commitment makes cross-context replay detectable at signature verification.

This follows the domain-separation pattern of TLS 1.3 key schedule [RFC8446] Section 7.1, where labels are

public constants providing domain separation without secrecy.

5.3. Stage External Properties

Stage G_1 (Golomb-Rice):
Achieves Shannon source coding bound for geometric distributions. PoT integer fields (timestamp delta, stratum, confidence) are geometrically distributed by construction (Poisson inter-arrival times, discretised). Output is byte-aligned. Complexity: O(n).

Stage R (Reed-Solomon(4,6) over GF(2^8)):
Achieves the Singleton bound as a Maximum Distance Separable (MDS) code. Any 4 of 6 shards reconstruct the original payload. Fixed-size equal shards output. Polynomial: x^8 + x^4 + x^3 + x^2 + 1.

Stage G_2 (Golay(23,12,7)):
Achieves the Hamming bound exactly as a perfect code: sum_{i=0}^{3} C(23,i) = 2048 = 2^{11}.
The unique non-trivial binary perfect code with t >= 2 correction (Tietavainen 1973). Corrects up to 3 bit errors per 23-bit block. Requires fixed-size input.

Stage H (HMAC-SHA256, 8-byte tag):
P(forge) <= 6 * 2^{-64} (union bound over 6 shards).
Public key by design (context separation, not secrecy).
Ed25519 seals GRG_Commitment: forging HMAC invalidates Ed25519 (double seal property).

5.4. Stage Ordering Rationale

The ordering G_1 -> R -> G_2 -> H is mathematically necessary. Any permutation degrades one or more provably tight properties:

G_1 before R (Theorem 1 of companion paper [POT2026]):
G_1 output is byte-aligned, providing GF(2^8)-optimal symbol boundaries for Reed-Solomon. Applying R before G_1 yields strictly greater RS parity overhead.

R before G_2 (Theorem 2 of companion paper):
Golay(23,12,7) requires fixed 23-bit input blocks. G_1 output is variable-length. R produces fixed-size equal shards, enabling zero-waste Golay encoding. RS and Golay provide orthogonal protection: P(fail) = P(RS) * P(Golay) < P(RS) + P(Golay).

G_2 before H (follows from Theorem 2):
HMAC seals the post-Golay shards. Ed25519 wraps grg_commitment (double seal).

These codes were selected for the same reason as deep-space missions: provably tight properties when retransmission is impossible. Golomb-Rice: JPL deep-space compression. Reed-Solomon: Cassini, Mars rovers. Golay(23,12,7): Voyager 1 and 2 Saturn images transmitted across 10^9 km (1980).

5.5. Verification Sequence

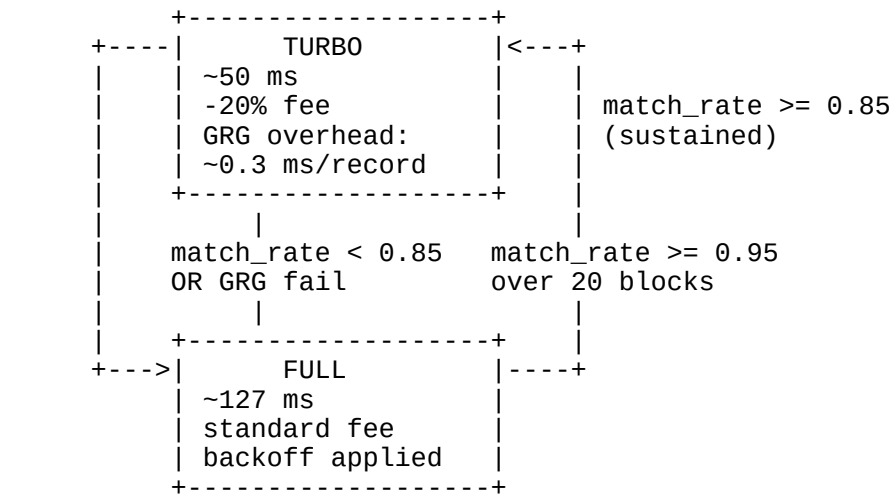
See Section 4.5. HMAC verification MUST precede Ed25519. This provides early rejection of context-invalid submissions at ~6 microseconds vs ~100 microseconds for Ed25519.

SECTION 6. ADAPTIVESWITCH

==========

6.1. State Machine

AdaptiveSwitch maintains per-node state in {TURBO, FULL}.



6.2. Transition Conditions and Hysteresis

TURBO entry:
   match_rate >= 0.95 sustained over >= 20 blocks.
   All PoT submissions within tier_tolerance.
   No GRG pipeline failures.

TURBO maintenance:
   match_rate >= 0.85 (relaxed threshold prevents flapping).

TURBO -> FULL:
   match_rate < 0.85 over any 20-block window, OR
   any GRG pipeline failure, OR
   any submission outside tier_tolerance (delay attack).

The hysteresis asymmetry is deliberate: trust is earned slowly and lost quickly (hard to earn, easy to lose).

6.3. Penalty and Exponential Backoff

On integrity failure in TURBO mode:
   Backoff penalty = 20 * 2^{f-1} blocks, maximum 320 blocks.
   (f = consecutive failure count)

On submission outside tier_tolerance:
   Immediate FULL mode transition.
   Backoff applies to TURBO re-entry.

6.4. Equilibrium Analysis (V* Threshold)

Let lambda = operator opportunity cost per millisecond.
Let c_0 = baseline ordering cost.
Let Delta_tau = 77 ms (TURBO vs FULL latency difference).

V* = c_0 + lambda * Delta_tau

For V < V*: ordering spam eliminated (E[S] = 0) in the unique symmetric Nash equilibrium.
For V >= V*: spam reduced by c_PoT / c_0 factor.

Empirical calibration from 151,423 Timeboost auctions (Arbitrum, April-July 2025):

Phase	lambda ($/ms)	V*	Result
Stable (May+)	0.11 - 0.23	$8.67	Spam eliminated
Central est.	0.16	$12.82	Spam eliminated
Competitive	1.13	$87.13	Spam eliminated
ETH L1 sandwich	--	($131)	Spam reduced

The Ethereum L1 average sandwich MEV ($131) lies above V*_max, consistent with "reduced but not eliminated" for highest-value attacks. For V < $8.67, PoT eliminates ordering manipulation entirely.

SECTION 7. TRANSPORT BINDING

7.1. TLS 1.3 via TLS Exporter Label

TTTPS uses the TLS Exporter mechanism [RFC5705] to derive PoT binding material from an established TLS 1.3 session, following the model of [RFC8915] Section 5.1.

This approach requires NO new TLS Extension Type codepoint and is fully backward-compatible with existing TLS 1.3 implementations. It resolves the codepoint collision risk of draft-helmprotocol-tttps-00 (0xFF50, Private Use range).

Exporter parameters:

```
Label:   "EXPORTER-tttps-pot-binding"
Context: PoT record bytes (all fields except Sig)
Length:  32 octets
```

Usage:

```
binding_key = TLS-Exporter("EXPORTER-tttps-pot-binding",
                           pot_record_without_sig,
                           32)
```

The binding_key MUST be used to verify that the PoT was generated within the current TLS session context. This prevents cross-session replay.

Verification procedure (normative):

The verifier MUST execute the following after TLS handshake completion and upon receiving a PoT frame:

```
expected_key = TLS-Exporter(
                 "EXPORTER-tttps-pot-binding",
                  pot_record_without_sig,
                  32)

If expected_key != binding_key_in_PoT_frame: REJECT.
```

The binding_key is carried in the first 32 octets of the PoT frame body, preceding the PoT record. Both client and server independently derive the same expected_key from the shared TLS session master secret. A PoT frame generated in session A cannot be replayed into session B because the TLS-Exporter output is session-specific (derived from the session’s master secret per RFC 5705 Section 4).

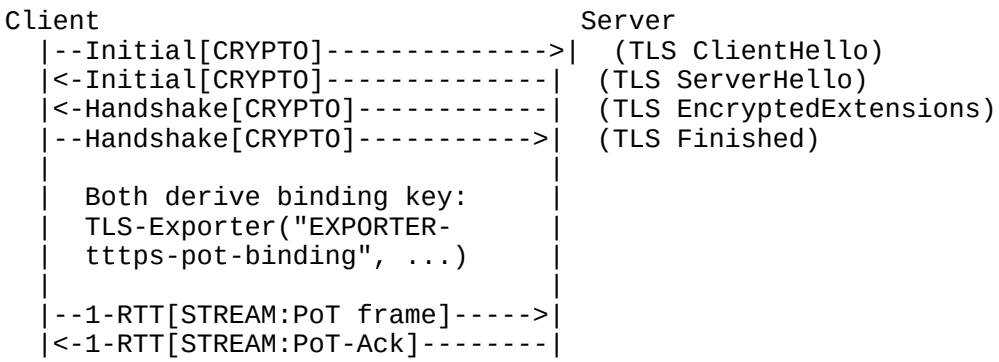
PoT Frame Extended Format (with binding):

```
   binding_key  (32 octets) -- TLS Exporter output
   pot_record   (143 octets) -- PoT record (Section 4.1)
   Total: 175 octets
```

7.2. QUIC Integration

TTTPS operates over QUIC [RFC9000] post-handshake.
The TLS Exporter is available after QUIC handshake completion.

QUIC + TTTPS flow:



PoT frames MUST be sent in a dedicated QUIC stream.
Stream type: 0x74 (defined in Section 11.4).

7.3. HTTP/3 Frame Type

Over HTTP/3 [RFC9114], PoT records are conveyed in a
dedicated HTTP/3 frame type.

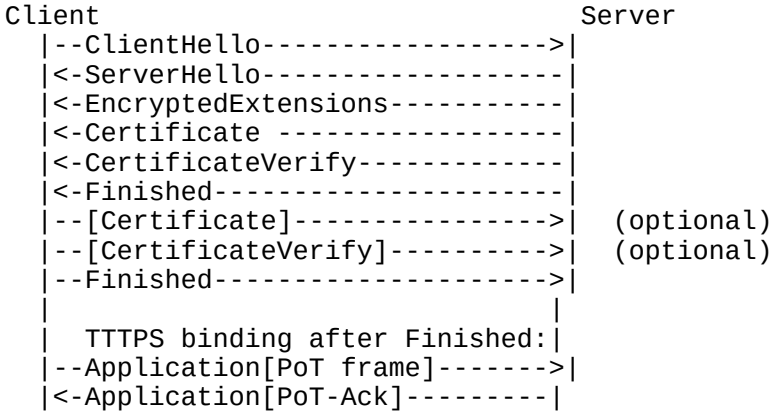
HTTP/3 PoT Frame format:

```
   Frame Type:   0x4C4F5400 (ASCII "LOT\0", IANA assigned,
                 see Section 11.4)
   Frame Length: variable (175 bytes with binding_key,
                 143 bytes for binding-free deployments)
   Frame Body:   PoT record (Section 4.1)
```

PoT frames MAY appear in any HTTP/3 request or response stream.
Servers MUST NOT reject requests solely on the basis of
absent PoT frames (backward compatibility).

7.4. Handshake Flow Diagrams

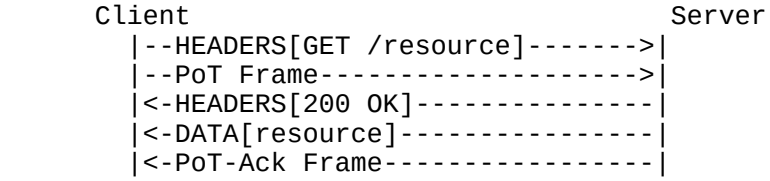
7.4.1. TLS 1.3 Flow



7.4.2. QUIC Flow

See Section 7.2.

7.4.3. HTTP/3 Flow



7.5. Backward Compatibility

Servers that do not implement TTTPS MUST be able to process TLS 1.3, QUIC, and HTTP/3 connections that include TTTPS binding material. TTTPS MUST NOT modify the TLS handshake in a way that causes negotiation failure with non-TTTPS peers.

Implementations SHOULD use ALPN [RFC7301] extension identifier "tttps/1" (IANA registration, Section 11) to negotiate TTTPS capability between peers.

SECTION 8. TIER STRUCTURE

TTTPS defines four time resolution tiers:

Tier	ID	Interval	Tolerance	Use Case
T0_epoch	0x0	6.4 min	60 s	Epoch ordering
T1_block	0x1	2 sec	2 s	L2 block (Base)
T2_slot	0x2	12 sec	12 s	L1 slot (ETH)
T3_micro	0x3	100 ms	100 ms	High-frequency trading
T-s1	0x4	—	3000 ms	Deep-space / Earth-Moon RTT

T-s1 Design Rationale:
Earth-Moon distance: ~384,400 km.
One-way propagation: c = 299,792 km/s → ~1,282 ms.
Earth-Moon RTT: ~2,600 ms.
T-s1 tolerance: 3,000 ms (400 ms headroom for processing and channel jitter). GRG(23,12,7) error correction tolerates deep-space bit-error rates without retransmission.

The code selection follows deep-space heritage: JPL Golomb-Rice compression, Reed-Solomon from Cassini and Mars rovers, and the Golay(23,12,7) perfect code used in the Voyager Saturn transmissions at 1.0e9 km.

Future tiers: 0x5 = T-s2 (Mars-Earth, 3-22 min), 0x6 = T-s3 (deep-space hours, store-and-forward). Values 0x5-0xF reserved.

Tier tolerance defines the maximum acceptable submission delay (submission_time - timestamp). Submissions outside tolerance trigger FULL mode per Section 6.3.

Fee discounts in TURBO mode are implementation-defined. Reference implementation: 20% discount [OPENTTT].

SECTION 9. SECURITY CONSIDERATIONS

9.1. Compromised NTP Sources and Path Attacks

This section addresses two threat models: compromised NTP servers and compromised network paths between the PoT Issuer and NTP servers (the standard IETF network adversary model).

Compromised NTP server:
A single compromised NTP source biases the synthesised timestamp by at most 1/k of the manipulation, where k >= 3 is the source count. For k=4 independent sources, single-source bias impact <= 0.25 of manipulation magnitude.

Compromised network path (IETF adversary model):
An attacker controlling the network path between the Issuer and one NTP server can inject delayed or replayed NTP responses. Two mitigations apply:

(a) Multi-source median: T_synth = median(T_1, ..., T_k) across k >= 3 sources from distinct administrative domains. A path-level attacker must simultaneously compromise paths to a majority of sources (e.g., both NIST and Cloudflare paths) to bias the median.

(b) Stratum tolerance check (Section 4.3, step 3): If max|T_i - T_synth| > stratum_tolerance, generation ABORTS. A single-path delay injection that pushes one source beyond tolerance is detected and rejected.

NTS [RFC8915] on the path between Issuer and NTP servers provides an additional layer of path authentication. Implementations SHOULD use NTS-authenticated sources where available.

Implementations MUST use sources from distinct administrative domains (e.g., NIST, Google, Cloudflare) to maximise independence. Sources from a single autonomous system MUST NOT be counted as independent.

9.2. Replay Prevention

Each PoT record includes a 256-bit cryptographically random Nonce (Section 4.1). Verifiers MUST maintain a nonce cache for the duration of the tier tolerance window. Duplicate nonces MUST be rejected.

The Ed25519 signature seals the nonce. Cross-session replay is additionally prevented by the TLS Exporter binding (Section 7.1).

9.3. Sybil Time Sources

An attacker controlling multiple NTP sources may attempt a Sybil attack on the synthesis median. The median is resistant to Sybil attacks when fewer than k/2 sources are compromised, for k >= 3. Implementations using k=4 are resistant to any single-source compromise.

9.4. Side-Channel Considerations

The HMAC verification (~6 microseconds) and Ed25519 verification (~100 microseconds) MUST be implemented in constant time. Variable-time implementations risk timing side-channel attacks against the HMAC key.

The Nonce MUST be generated with a constant-time CSPRNG.

9.5. Byzantine Economic Attacks

An attacker may attempt to manipulate ordering for economic gain (MEV). The AdaptiveSwitch V* threshold (Section 6.4) ensures that for V < V*_min = $8.67, ordering spam is eliminated in the unique Nash equilibrium.

Attackers with V >= V*_max = $87.13 may find manipulation economically rational at the margin. PoT reduces expected spam for such cases by a factor of c_PoT / c_0 (Section 6.4).

9.6. Delay-Based Temporal Attacks

[RFC8915] Section 8.6 identifies delay attacks as a primary threat to time synchronisation security, noting that an adversary who delays NTP packets can cause a client to accept a stale timestamp as current.

TTTPS addresses this threat through two complementary gates, applied in sequence at verification time (Section 4.5):

(1) HMAC context gate (Section 5.2, ~6 microseconds):
A PoT generated at time T with context ctx_id cannot be presented in a different execution context ctx_id' without HMAC verification failing. Context includes chain_id and pool_address. An attacker cannot reuse a valid PoT from a previous context window.

This is analogous to the cookie freshness mechanism of [RFC8915] Section 5.4, which binds cookies to TLS session keys that expire with the session.

(2) AdaptiveSwitch recency gate (Section 6.3):
A PoT submitted at time S where (S - T) > tier_tolerance is treated as a conformance failure. FULL mode is triggered immediately. The submission is rejected regardless of cryptographic validity.

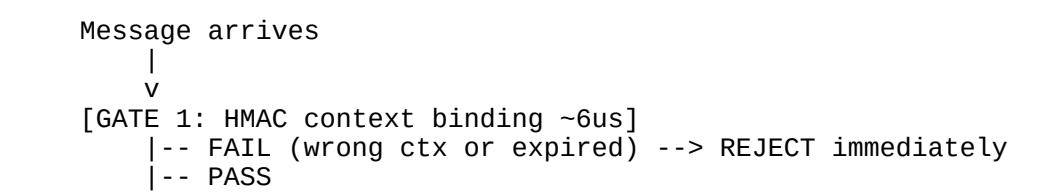
This reflects the operational observation, consistent with [RFC8915] Section 8.6, that in correctly functioning networks legitimate submissions arrive within tier tolerance bounds. Submissions outside this window correlate with Byzantine behaviour.

FILO+GRG processing discipline:
Among PoT records that pass both gates, the most recently generated qualifying submission is processed first. This creates an adverse incentive structure for delay attackers: a delayed-but-valid PoT that bypasses the HMAC gate is rejected at the recency gate; a PoT that passes both gates competes at a recency disadvantage against honest peers.

Together, these mechanisms render delay-based attacks economically irrational:

- A delayed PoT fails the recency gate -> FULL mode
- Repeated FULL mode triggers exponential backoff
- Backoff cost exceeds MEV opportunity for V < V*

FILO+GRG flow:



```
        v
    [GATE 2: AdaptiveSwitch recency check]
        |-- FAIL (submission > tier_tolerance) --> FULL mode
        |-- PASS
        v
    [Enter FILO processing queue]
        |
        v
    Most recently generated qualifying PoT processed first
```

9.7. GRG Pipeline Security

Byzantine context binding provides:
P(detect Byzantine manipulation) >= 1 - 2^{-61}

This follows from: P(forge_i) = 2^{-64} per shard (PRF security of HMAC [Bellare1996]); union bound over 6 shards: P(forge all) <= 6 * 2^{-64} < 2^{-61} ~= 4.3e-19.

This transforms SCCP from P(detect) < 1 (Shannon model) to P(detect) >= 1 - 2^{-61}.

Implementations MUST NOT expose GRG internal state, shard values, or intermediate pipeline results through public APIs or error messages.

9.8. AEAD Protection of TTTPS Metadata

When TTTPS operates as a layer alongside TLS (Section 1.2, Phase 1), a structural gap exists: TLS encrypts and authenticates the application payload, but the TTTPS header fields (timestamp, GRG commitment, tier, sequence number) are conveyed outside the TLS record layer.

An attacker who can observe or modify QUIC/UDP datagrams may tamper with TTTPS metadata without breaking TLS. Specific attacks include:

- Timestamp substitution: replace a PoT timestamp field to shift apparent ordering without invalidating the TLS session (the TLS layer does not verify TTTPS fields).

- Replay with modified tier: downgrade a T3_micro PoT (100ms tolerance) to T0_epoch (60s tolerance) to evade the recency gate.

Mitigation: AEAD (ChaCha20-Poly1305) seals all TTTPS header fields together with the payload under a session key derived from the Ed25519 handshake. This closes the gap: tampering with any TTTPS field causes AEAD.open to fail at ~1 μs, before any further processing.

Implementations SHOULD derive the AEAD session key as:

```
session_key = HKDF-SHA256(
                ed25519_session_secret,
                "tttps-session-aead-v1",
                32)
```

After session establishment via Ed25519 (once per connection), all subsequent PoT frames are protected by AEAD per-packet. Ed25519 is NOT repeated per packet; this matches the TLS 1.3 model of asymmetric authentication at handshake, symmetric encryption at record layer.

SECTION 10. PRIVACY CONSIDERATIONS

10.1. Unlinkability

PoT records include a 256-bit random Nonce (Section 4.1) that MUST be freshly generated for each record. This prevents linkage of PoT records from the same issuer across sessions.

The TLS Exporter binding (Section 7.1) ensures that PoT records are bound to specific TLS sessions and cannot be used to correlate activity across sessions.

Issuers SHOULD NOT include in PoT records any information beyond the fields defined in Section 4.1 that could enable participant identification.

10.2. Minimal Disclosure

The PoT wire format (Section 4.1) does not include:

- Participant identity or address
- Transaction content
- Economic parameters or bid values

The ctx_id (chain_id || pool_address) is a public identifier already disclosed by the on-chain context. Its inclusion in the HMAC key does not introduce new disclosures.

SECTION 11. IANA CONSIDERATIONS

11.1. TLS Exporter Labels Registry

IANA is requested to add the following entry to the "TLS Exporter Labels" registry [RFC5705]:

Label:	EXPORTER-tttps-pot-binding
DTLS-OK:	Y
Recommended:	Y
Reference:	[this document] Section 7.1

11.2. TTTPS Tier Registry

IANA is requested to create a new registry "TTTPS Tier Identifiers" with the following initial values:

Value	Name	Interval	Reference
0x0	T0_epoch	6.4 min	[this document]
0x1	T1_block	2 sec	[this document]
0x2	T2_slot	12 sec	[this document]
0x3	T3_micro	100 ms	[this document]
0x4	T-s1	3000 ms	[this document]
0x5-F	Reserved	--	Specification Required

Registration procedure: Specification Required.

11.3. Time Source Type Registry

IANA is requested to create a new registry "TTTPS Time Source Types" with the following initial values:

Value	Name	Reference
0x01	NIST	[this document]
0x02	Google	[this document]
0x03	Cloudflare	[this document]
0x04	Apple	[this document]
0x05-FE	Unassigned	Specification Required
0xFF	Private Use	[this document]

11.4. HTTP/3 and QUIC Stream Types

IANA is requested to add the following entries:

HTTP/3 Frame Types registry:
- Type: TBD (to be assigned by IANA; 0x4C4F5400 proposed)
- Name: TTTPS_POT_FRAME
- Reference: [this document] Section 7.3
- Note: Implementations MUST use the IANA-assigned value.

QUIC Stream Types registry:
- Type: TBD (to be assigned by IANA)
- Name: TTTPS_POT_STREAM
- Reference: [this document] Section 7.2
- Note: Implementations MUST use the IANA-assigned value. Until assigned, use 0x74 for testing only.

11.5. PoT Extension Type

TTTPS-00 referenced TLS Extension Type 0xFF50 (Private Use range). TTTPS-01 does NOT use a TLS Extension Type. The TLS Exporter mechanism (Section 7.1) requires no codepoint.

If a future version requires a TLS Extension Type, the authors will request a codepoint via Specification Required procedure per [RFC8447].

SECTION 12. INTELLECTUAL PROPERTY

The GRG Integrity Pipeline is covered by pending patent applications filed by the authors. Full specification of the GRG pipeline, including stage implementations and parameter selection, will be made available upon conclusion of patent proceedings (targeted Q3 2026).

In accordance with IETF policy [BCP79], the authors are prepared to license any patents essential to this specification on FRAND terms.

Independent implementation is possible using:
- The abstract interface in Section 5.1
- The external properties in Section 5.3
- The reference implementation [OPENTTT]

This follows the precedent of [RFC8915] Section 6, which specifies cookie format as implementation-dependent.

SECTION 13. REFERENCES

13.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, March 2010.

[RFC7301] Friedl, S. et al., "TLS Application-Layer Protocol Negotiation Extension", RFC 7301, July 2014.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, August 2018.

[RFC8126] Cotton, M. et al., "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, June 2017.

[RFC8447] Salowey, J. and S. Turner, "IANA Registry Updates for TLS and DTLS", RFC 8447, August 2018.

[RFC8915] Franke, D. et al., "Network Time Security for the Network Time Protocol", RFC 8915, September 2020.

[RFC9000] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, May 2021.

[RFC9114] Bishop, M., "HTTP/3", RFC 9114, June 2022.

13.2. Informative References

[Bellare1996] Bellare, M., Canetti, R., and H. Krawczyk, "Keying Hash Functions for Message Authentication", CRYPTO 1996, LNCS 1109, 1996.

[Bernstein2012] Bernstein, D.J. et al., "High-speed high-security signatures", J. Cryptogr. Eng. 2, 77-89, 2012.

[Castro1999] Castro, M. and B. Liskov, "Practical Byzantine Fault Tolerance", OSDI, 173-186, 1999.

[EIGENPHI] EigenPhi Research, "MEV sandwich attacks: annual loss estimates", 2025.

[FLASHBOTS] Flashbots, "MEV explore", 2025. https://explore.flashbots.net

[Golomb1966] Golomb, S.W., "Run-length encodings", IEEE Trans. Inf. Theory 12, 399-401, 1966.

[Mazorra2026] Mazorra, B., Schmid, L. and D. Tse, "Timing games: probabilistic backrunning and spam", arXiv:2602.22032, 2026.

[Messias2025] Messias, J. and C.F. Torres, "The express lane to spam and centralization: an empirical analysis of Arbitrum’s Timeboost", arXiv:2509.22143, 2025.

[OPENTTT] Helm Protocol, "OpenTTT SDK", https://github.com/Helm-Protocol/OpenTTT, npm install openttt, 2026.

[POT2026] Jorgen, H., "Proof-of-Time: Byzantine-Resilient Temporal Ordering in Untrusted Networks", March 2026. IETF draft-helmprotocol-tttps-00.

[Reed1960] Reed, I.S. and G. Solomon, "Polynomial codes over certain finite fields", SIAM J. Appl. Math. 8, 300-304, 1960.

[Tietavainen1973] Tietavainen, A., "On the nonexistence of perfect codes over finite fields", SIAM J. Appl. Math. 24, 88-96, 1973.

[GLASSWING] Anthropic, "Project Glasswing: Securing Critical Software for the AI Era", https://www.anthropic.com/project/glasswing, April 2026.

[MAZORRA2026note] Jorgen, H., "Proof-of-Time: Completing the Timing Game", The Flashbots Collective, https://collective.flashbots.net/t/proof-of-time-completing-the-timing-game/5633, March 2026.

[Zhang2026] Zhang, J. et al., "Hyperagents: self-referential agents with metacognitive self-modification", arXiv:2603.19461, 2026.

APPENDIX A. ADAPTIVESWITCH TLA+ SPECIFICATION

The following TLA+ module formally specifies the AdaptiveSwitch state machine. The module is verified by the TLC model checker with parameters MaxNodes=3, MaxBlocks=10, TierToleranceMs=100, TurboEntry=95, TurboMaintain=85.

The module specifies:

- TypeInvariant: all five state variables are well-typed.
- S1 (NoForcedTurbo): TURBO requires match_rate >= 85 AND fail_count = 0 -- conjunction, not disjunction.
- S2 (DelayRejectionTriggersFull): submission outside tier tolerance is incompatible with TURBO.
- S3 (FailureExcludesTurbo): any integrity failure forces FULL.
- L1 (EventualTurbo): a node with sustained good behaviour eventually reaches TURBO (liveness under weak fairness).

EnvStep models the environment updating match_rate, fail_count, and submission_delay nondeterministically, ensuring the invariants hold under all adversarial input sequences.

The module structure follows the AgentLifecycle pattern of Helm Autonomy Layer Yellow Paper v2.0 [HelmYP2026].

```
---- MODULE AdaptiveSwitch ----
EXTENDS Naturals, FiniteSets

CONSTANTS MaxNodes, MaxBlocks, TurboEntry, TurboMaintain,
          TierToleranceMs

ASSUME /\ TurboEntry    = 95   \* 95%  match_rate required for TURBO
       /\ TurboMaintain = 85   \* 85%  minimum to stay in TURBO
```

```
       /\ TierToleranceMs > 0  \* positive tier tolerance (ms)

NodeId == 1..MaxNodes   \* finite set of node identifiers
Modes  == { "TURBO", "FULL" }

VARIABLES
  node_mode,        \* [NodeId -> Modes]  per-node state
  match_rate,       \* [NodeId -> 0..100] ordering-match percentage
  fail_count,       \* [NodeId -> Nat]    consecutive integrity failures
  block_count,      \* Nat                current block number
  submission_delay  \* [NodeId -> Nat]    ms since last PoT generation

vars == <<node_mode, match_rate, fail_count,
          block_count, submission_delay>>

\* ——— Helpers ——————————————————————————————————————————————————————————
————————————————————————
SubmittedOutsideTolerance(n) ==
  submission_delay[n] > TierToleranceMs

\* ——— Type correctness ——————————————————————————————————————————————————
————————————————
TypeInvariant ==
  /\ node_mode        \in [NodeId -> Modes]
  /\ match_rate       \in [NodeId -> 0..100]
  /\ fail_count       \in [NodeId -> Nat]
  /\ block_count      \in Nat
  /\ submission_delay \in [NodeId -> Nat]

\* ——— Initial state (all nodes start in FULL, zero counters) ————————————
Init ==
  /\ node_mode        = [n \in NodeId |-> "FULL"]
  /\ match_rate       = [n \in NodeId |-> 0]
  /\ fail_count       = [n \in NodeId |-> 0]
  /\ block_count      = 0
  /\ submission_delay = [n \in NodeId |-> 0]

\* ——— Actions ———————————————————————————————————————————————————————————
————————————————————

\* Promote n from FULL to TURBO when match_rate sufficient
\* and no pending failures.
PromoteToTurbo(n) ==
  /\ node_mode[n]   = "FULL"
  /\ match_rate[n]  >= TurboEntry
  /\ fail_count[n]  = 0
  /\ ~SubmittedOutsideTolerance(n)
  /\ node_mode' = [node_mode EXCEPT ![n] = "TURBO"]
  /\ UNCHANGED <<match_rate, fail_count,
                 block_count, submission_delay>>

\* Demote n from TURBO to FULL on poor match_rate, integrity
\* failure, or submission outside tier tolerance.
DemoteToFull(n) ==
  /\ node_mode[n] = "TURBO"
  /\ \/ match_rate[n] < TurboMaintain
     \/ fail_count[n] > 0
     \/ SubmittedOutsideTolerance(n)
  /\ node_mode' = [node_mode EXCEPT ![n] = "FULL"]
  /\ UNCHANGED <<match_rate, fail_count,
                 block_count, submission_delay>>

\* Environment step: update match_rate / fail_count / delay
\* (models external inputs; unconstrained for model checking)
EnvStep(n, mr, fc, sd) ==
  /\ match_rate'       = [match_rate       EXCEPT ![n] = mr]
```

```
  /\ fail_count'       = [fail_count       EXCEPT ![n] = fc]
  /\ submission_delay' = [submission_delay EXCEPT ![n] = sd]
  /\ block_count'      = block_count + 1
  /\ UNCHANGED node_mode

Next ==
  \E n \in NodeId :
    \/ PromoteToTurbo(n)
    \/ DemoteToFull(n)
    \/ \E mr \in 0..100, fc \in 0..5, sd \in 0..(TierToleranceMs+50) :
         EnvStep(n, mr, fc, sd)

Spec == Init /\ [][Next]_vars /\ WF_vars(Next)

\* ——— Safety invariants ———————————————————————————————————————————
———————————————

\* S1: TURBO requires healthy match_rate AND no integrity failures.
NoForcedTurbo ==
  \A n \in NodeId :
    node_mode[n] = "TURBO" =>
      /\ match_rate[n] >= TurboMaintain
      /\ fail_count[n] = 0

\* S2: Delay outside tier tolerance must not coexist with TURBO.
DelayRejectionTriggersFull ==
  \A n \in NodeId :
    SubmittedOutsideTolerance(n) => node_mode[n] = "FULL"

\* S3: fail_count > 0 must not coexist with TURBO.
FailureExcludesTurbo ==
  \A n \in NodeId :
    fail_count[n] > 0 => node_mode[n] = "FULL"

\* ——— Liveness ————————————————————————————————————————————————————
———————————————————

\* L1: A node with sustained good behaviour eventually reaches TURBO.
EventualTurbo ==
  \A n \in NodeId :
    (match_rate[n] >= TurboEntry /\ fail_count[n] = 0
      /\ ~SubmittedOutsideTolerance(n))
    ~> node_mode[n] = "TURBO"

\* ——— TLC model values (for model checking) ————————————————————————
—
\* MaxNodes = 3, MaxBlocks = 10, TierToleranceMs = 100
\* TurboEntry = 95, TurboMaintain = 85
====

   The invariant NoForcedTurbo corresponds to Safety Property S4
   of Helm Yellow Paper v2.0 (AS score external immutability).


============================================================
APPENDIX B.  GRG PIPELINE SPECIFICATION (PLACEHOLDER)
============================================================

   The GRG Integrity Pipeline (Section 5) processes PoT payloads
   through four stages: Golomb-Rice (G_1), Reed-Solomon (R),
   Golay(23,12,7) (G_2), and HMAC-SHA256 (H).

   The stage ordering G_1 -> R -> G_2 -> H is mathematically
   necessary, as proven in [POT2026] Theorems 1-3.

   Full specification of internal cryptographic operations,
```

parameter selection, and implementation details will be
published upon conclusion of pending patent proceedings
(targeted Q3 2026).

Reference implementation: https://github.com/Helm-Protocol/OpenTTT
npm: npm install openttt

Independent implementations of the abstract interface
(Section 5.1) and external properties (Section 5.3) are
permitted under BSL-1.1 license terms.

APPENDIX C. TEST VECTORS

Test vectors for PoT generation and verification are provided
as property-based tests rather than deterministic byte vectors.
This approach prevents reverse-engineering of GRG pipeline
parameters from expected byte sequences.

Required properties (all MUST pass):

C.1 Lossless round-trip:
GRG_Inverse(GRG(P, ctx)) = P for all P, ctx

C.2 Nonce uniqueness:
Two calls to Generate() MUST NOT produce equal Nonces.

C.3 Context separation:
GRG(P, ctx_A) != GRG(P, ctx_B) for ctx_A != ctx_B
(negligible probability of collision: < 2^{-61})

C.4 Verification correctness:
Verify(Generate(P, ctx), ctx) = TRUE

C.5 Forgery resistance:
Verify(tampered_record, ctx) = FALSE for any
single-bit modification to P or GRG_Commitment.

C.6 Delay rejection:
A PoT submitted at T + tier_tolerance + 1ms MUST
trigger FULL mode.

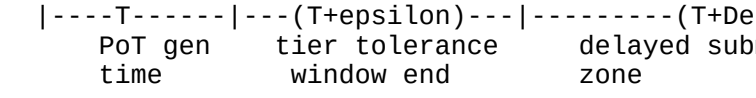
C.7 HMAC gate priority:
HMAC verification MUST complete before Ed25519 is
attempted. Invalid HMAC MUST NOT result in Ed25519
invocation (measurable via timing).

Reference test suite: 365 tests, 31 suites, 100% pass rate
[OPENTTT]. The test suite uses property-based testing only
(no deterministic byte vectors).

APPENDIX D. FILO+GRG DELAY REJECTION FLOW

This appendix provides a normative ASCII diagram of the
FILO+GRG delay rejection mechanism described in Section 9.6.

TIME AXIS:



```
   VALID SUBMISSION WINDOW: [T, T + tier_tolerance]
   DELAYED ZONE: (T + tier_tolerance, infinity)

   GATE 1: HMAC context binding (~6 microseconds)
   ------------------------------------------------
   Input: PoT record + submission context

   If HMAC(k, shard_i) does not match for any i:
     -> REJECT immediately
     -> DO NOT invoke Ed25519
     Covers: wrong context, tampered payload

   GATE 2: AdaptiveSwitch recency check
   ------------------------------------------------
   Input: PoT record + current submission time S

   If (S - PoT.Timestamp) > tier_tolerance:
     -> REJECT
     -> Trigger FULL mode
     -> Apply exponential backoff
     Covers: valid PoT submitted outside tolerance window

   FILO QUEUE (Gate 1 AND Gate 2 passed)
   ------------------------------------------------
   Queue discipline: most recently generated PoT first.

   If multiple PoTs qualify:
     Select max(PoT.Timestamp) for processing.
     Earlier PoTs remain in queue.

   Effect on delay attackers:
   o  Cannot pass Gate 2 (recency check rejects)
   o  Even if somehow past Gate 2, lose priority to fresher PoTs
   o  Repeated failures -> exponential backoff -> self-defeating

   COMPLEXITY NOTE:
   Gate 1 (HMAC): O(1) per record, ~6 microseconds
   Gate 2 (recency): O(1) per record, ~0.1 microseconds
   Queue ordering: O(log q) for q queued records (priority queue)
   Total per-record: O(1) -- independent of network size n

   Compare with BFT consensus protocols:
     PBFT/Tendermint/HotStuff: O(n^2) network-wide message
     exchanges to reach Byzantine TOLERANCE (tolerate f < n/3
     Byzantine nodes) for n total nodes.

   TTTPS achieves Byzantine ELIMINATION at O(1) per record.
   Honest user verification cost: ~106 microseconds (constant).
   Attacker economic cost: increases as V* threshold makes
   manipulation irrational (E[profit] < 0 for V < $8.67).
   Attacker backoff cost: O(2^f) blocks for f failures.
   Network scaling: 100 nodes -> 1,000,000 nodes: BFT cost
   grows 10^8x; TTTPS per-record cost unchanged.

============================================================
END OF DRAFT
============================================================
```

Acknowledgements

The authors thank the IETF dispatch list reviewers (Ekr, Worley, Jim, Tim) for feedback on draft-helmprotocol-tttps-00. The GRG pipeline selection rationale builds on deep-space engineering heritage: JPL Golomb-Rice compression, RS codes from Cassini and the Mars rovers, and Golay(23,12,7) from the Voyager Saturn transmissions (1.0e9 km, 1980).

Author’s Address

Heime Jorgen
Kenosian
Email: heime.jorgen@proton.me
IETF Draft: draft-helmprotocol-tttps-02