Network Working Group
Internet-Draft
Intended status: Informational
Expires: 20 November 2026

A. Mery
K. G. Nagy
Apptly Software Ltd

19 May 2026

Taistamp: Signed TAI64N Timestamps over HTTP
draft-mery-nagy-taistamp-00

Abstract

This memo defines Taistamp, an HTTP-accessible service that returns the current time as a TAI64N label, optionally signed with Ed25519. The service is reached at the well-known URI /.well-known/taistamp and is intended for clients that need an authenticated wall-clock reading without running an NTP stack or relying on the client's own clock for TLS certificate validation. The signing scheme uses a length-prefixed, domain-separated framing and publishes verification keys as DNS TXT records in a DKIM-inspired selector layout.

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). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on 20 November 2026.

Copyright Notice

Table of Contents

1. Introduction

Many small clients need an accurate wall-clock reading to validate certificates, evaluate signature freshness, or stamp local events. Two common sources fall short:

* An NTP client is intrusive on constrained or sandboxed runtimes and exposes the device to a separate trust domain.
* TLS does not provide an explicit authenticated time value to the client; certificate validity checks depend on the client's existing clock rather than establishing a trustworthy time reading.

Taistamp addresses these by serving the current time as a 25-byte TAI64N [TAI64] label over HTTP, with an optional Ed25519 [RFC8032] signature bound to a client-supplied nonce. Verification keys are published in DNS as TXT records under a DKIM-style [RFC6376] selector, so keys can be rotated without coordinating with clients.

The protocol is deliberately small. A compliant implementation fits in a single HTTP request handler, has no state beyond its signing key, and is independent of any specific HTTP framework or runtime.

2. Design Rationale

2.1. Why HTTP

HTTP is available everywhere a network connection is. Browsers, serverless runtimes (Cloudflare Workers, AWS Lambda, Deno Deploy), edge nodes, and CDN origins can all serve and consume HTTP without additional infrastructure. Port 443 (or 80) is open where UDP 123 (NTP), UDP 2002 (Roughtime, port assignment pending), and UDP 4014 (TAICLOCK [TAICLOCK]) are commonly firewalled. A compliant Taistamp server requires nothing beyond a single request handler; a compliant client requires nothing beyond an HTTP library, both of which exist in every language and runtime in common use today.

Taistamp works over plain HTTP. The Ed25519 signature provides authentication and integrity at the application layer, independent of the transport: any in-transit modification breaks the signature, and a client-supplied nonce prevents replay. These guarantees apply only to responses bearing a valid TAI-Signature matching a request TAI-Nonce. A network attacker who cannot forge the signature can at most drop or delay responses, which is a denial of service rather than a security breach.

TLS [RFC8446] is RECOMMENDED. It adds transport confidentiality (hiding that a client is querying for time and from which server) and an additional layer of integrity. Where used, the two layers are complementary: TLS protects the channel, the Ed25519 signature protects the claim independently of which certificate authority issued the server certificate.

The server is stateless beyond its signing key. This is a deliberate design goal: a stateless handler can be deployed as a CDN edge function, a serverless worker, or behind a load balancer with no session affinity or shared state.

Taistamp is transported over TCP-based HTTP, which rules out UDP-style spoofed-source amplification. The protocol-mandated fields of a signed response contribute at most approximately 530 bytes at maximum field lengths, excluding transport framing and any headers added by intermediaries — the response body is a fixed 25 bytes, and the protocol-defined headers (TAI-Nonce echo, TAI-Signature, TAI-Key-Selector, TAI-Leap-Seconds) account for the remainder — an attacker cannot induce a response meaningfully larger than the request, so Taistamp is not usable as an amplification vector.

2.2. Why DNS-Based Key Discovery

Roughtime [Roughtime] addresses the same authenticated time problem but distributes trust differently: clients ship with a configured list of trusted server public keys. Key rotation therefore requires a client update or a coordinated key-distribution mechanism outside the protocol.

Taistamp operators instead publish verification keys as DNS TXT records under a DKIM-inspired [RFC6376] selector scheme; they rotate a key by publishing a new TXT record under a new selector and reconfiguring the server, with no client coordination required. A client that has cached a signed response can continue to verify it as long as the TXT record at the old selector remains in DNS. DNSSEC [RFC4033] protects the key material end-to-end and is strongly recommended where the operator controls a signed zone.

This approach deliberately trades the ecosystem-level misbehaviour detection that Roughtime’s multi-server cross-checking provides for simpler, decentralised deployability: any operator with a domain name and an HTTP endpoint can run a Taistamp server without registering with a central authority.

2.3. Why TAI64N

POSIX time and UTC both admit leap seconds, making a given second ambiguous: a timestamp of 23:59:60 is legal in UTC but meaningless in most system clocks, which either smear the leap or repeat the last second. For applications that check certificate validity windows or evaluate signature freshness, an ambiguous second is a correctness hazard.

TAI [TAI64] is strictly monotonic. TAI64N encodes TAI seconds and nanoseconds in a fixed 25-byte ASCII label with no ambiguity at any leap-second boundary. The TAI-Leap-Seconds header carries the offset as informational guidance; clients requiring UTC equivalence MUST source the offset from a trusted table rather than from the server (see Section 10.6).

2.4. Why the External Representation

TAICLOCK [TAICLOCK] encodes timestamps as 16 bytes of binary TAI64NA (8 bytes of seconds, 4 of nanoseconds, 4 of attoseconds). Taistamp instead uses the TAI64N external representation [TAI64]: the ASCII character @ followed by 16 lowercase hexadecimal digits for the seconds field and 8 for the nanoseconds field, totalling 25 bytes.

The 9-byte difference over TAICLOCK’s binary encoding arises from three sources: the @ leader (+1 byte), hex-expanding the 8-byte seconds field to 16 ASCII digits (+8 bytes), and hex-expanding the 4-byte nanoseconds field to 8 ASCII digits (+4 bytes), offset by dropping the 4-byte attosecond field that TAI64NA carries and TAI64N omits (−4 bytes).

The choice is driven by the HTTP context. A hex-encoded ASCII label is trivially inspectable with curl, safe to log as plain text, and requires no additional encoding in an HTTP response body. The 9-byte overhead over binary is negligible in an HTTP exchange. Attoseconds are omitted because no deployed clock provides sub-nanosecond resolution.

2.5. Why Ed25519

Ed25519 [RFC8032] was chosen for three reasons. First, verification is fast and keys and signatures are compact: a public key is 32 bytes and a signature is 64 bytes, keeping the signed response small and the verification cost negligible on constrained clients. Second, signing is deterministic — no per-signature random nonce is required, which removes an entire class of implementation hazard present in ECDSA. Third, Ed25519 has no negotiable parameters; there is nothing to downgrade.

2.6. Why Nonce-Based Anti-Replay

An alternative to a client-supplied nonce would be a server-enforced freshness window: reject any response older than N seconds. This approach is circular: it requires the client to have an approximately correct clock already, which is precisely the precondition Taistamp is designed to avoid.

A client-supplied nonce breaks the circularity. The client need have no prior knowledge of the time; it generates an unpredictable value, sends it in the request, and the server's signature binds the returned label to that value. A replayed response from an earlier exchange will carry a different nonce and the signature will not verify.

2.7. Precision Expectations

The TAI64N format encodes nanoseconds, but server implementations are commonly limited to millisecond resolution. Runtimes such as Cloudflare Workers deliberately freeze the clock at the last I/O boundary as a Spectre mitigation; no sub-millisecond source is available. Clients SHOULD NOT assume that the nanosecond field carries sub-millisecond information; in practice the last six decimal digits of the nanosecond value are often zero.

2.8. Non-Goals

Taistamp does not synchronise or discipline a local clock; callers MUST treat the returned label as a single measurement subject to network round-trip uncertainty. It does not protect against a server operator who deliberately signs an incorrect time: the signature attests to what the server claimed, not to the accuracy of its clock. It does not provide confidentiality; TLS [RFC8446] SHOULD be used when confidentiality of the query or response is required. It is not a document timestamping service in the sense of RFC 3161 [RFC3161]: it timestamps moments, not artefacts.

3. Relation to Other Protocols

TAICLOCK [TAICLOCK] is D. J. Bernstein's UDP-based TAI64N time service (IANA port 4014). A client sends a packet between 20 and 256 bytes whose first four bytes are the ASCII string ctai; the server echoes the packet with the first byte replaced by s and bytes 419 replaced with the current time in TAI64NA format. TAICLOCK is the direct conceptual predecessor of Taistamp: both serve TAI64N timestamps on demand. Taistamp differs in using HTTP (deployable over existing HTTPS infrastructure, no dedicated daemon), adding optional Ed25519 signatures, and publishing keys via DNS rather than

relying on the transport for integrity.

RFC 5905 (NTP) [RFC5905] synchronises a local clock continuously over UDP, adjusting for network delay across multiple exchanges. Taistamp makes no attempt to discipline a local clock; it provides a single authenticated reading that the caller treats as a measurement subject to one round-trip time.

RFC 8915 (NTS) [RFC8915] adds cryptographic authentication to NTP using TLS for key exchange and AEAD for packet authentication. NTS inherits NTP's requirement for a dedicated daemon and UDP port 123. Taistamp targets runtimes where neither is available.

RFC 3161 (TSP) [RFC3161] is a document timestamping protocol: a client submits a hash of a document and receives a signed token attesting that the document existed at a given time. TSP timestamps artefacts; Taistamp timestamps moments. The two protocols are complementary rather than overlapping.

Roughtime [Roughtime] is an authenticated time protocol that also uses Ed25519 signatures and is designed for constrained clients. It operates over UDP, distributes trust via a configured list of server public keys, and detects misbehaving servers by comparing timestamps and accuracy radii across multiple providers. Taistamp is simpler and more narrowly scoped: a single HTTP endpoint, DNS-based key discovery, and no ecosystem coordination requirement.

4. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The terms "TAI" and "TAI64N" are used as defined in [TAI64].

The term "selector" is used in the DKIM sense [RFC6376]: an opaque, host-scoped label that selects one verification key from possibly many published under the same host.

The term "client" refers to any HTTP agent that fetches the Taistamp resource at /.well-known/taistamp.

The term "verifier" refers to a client that performs signature verification on a Taistamp response.

HTTP field values defined in this document use Structured Field Values [RFC9651]. Type annotations (Integer, Binary, Token) refer to the item types defined in that specification.

5. Resource

A Taistamp service is reached at the well-known URI [RFC8615]:

/.well-known/taistamp

A request URI MUST be the exact path above. Query strings, if any, are ignored by this version of the protocol and MUST NOT change the response.

5.1. Methods

A server MUST accept GET and HEAD. A server that supports CORS (see Section 5.2) MUST also accept OPTIONS; a server without CORS support MAY accept OPTIONS. A server MUST respond 405 Method Not Allowed to any method it does not accept. The Allow header on a 405 response, and on a successful OPTIONS response, MUST enumerate the methods the server accepts ([RFC9110], Section 10.2.1): a server that accepts OPTIONS emits Allow: GET, HEAD, OPTIONS; one that does not emits Allow: GET, HEAD.

A HEAD response MUST NOT include a body and MUST carry the same headers as the corresponding GET response, with the exception that TAI-Nonce, TAI-Key-Selector, and TAI-Signature MUST NOT be present. Because the signed payload covers the response body, a signature cannot be verified without it; HEAD is therefore useful only as a liveness or configuration probe. Note that this makes HEAD responses non-equivalent to the corresponding GET response in the sense of [RFC9110] Section 9.3.2; carrying an unverifiable signature on a bodyless response would be misleading.

OPTIONS responses MUST NOT carry any TAI-* response header and MUST NOT be signed.

5.2. Cross-Origin Resource Sharing

A Taistamp server intended for access from browser pages SHOULD respond to OPTIONS requests with status 200 and the following headers:

* Access-Control-Allow-Origin: * (or a site-specific origin policy)
* Access-Control-Allow-Methods: GET, HEAD
* Access-Control-Allow-Headers: TAI-Nonce

* Access-Control-Expose-Headers: TAI-Leap-Seconds, TAI-Nonce, TAI-Key-Selector, TAI-Signature
* Access-Control-Max-Age: 600 (or a higher operator- chosen value)

A browser making a cross-origin request that includes the TAI-Nonce header will send an OPTIONS preflight before the actual GET; without the appropriate Access-Control-* response headers the browser will block the request. Non-browser clients are unaffected.

Servers SHOULD emit Access-Control-Max-Age with a value of at least 600 seconds. Pre-flight responses carry no per-request data and are safe to cache for the lifetime of the deployed CORS configuration. In the absence of this header, browser pre-flight cache lifetimes diverge significantly across implementations, which may cause a pre-flight request to precede every cross-origin GET in high-traffic deployments.

5.3. Successful Response

A successful GET response has status 200 OK and a body consisting of exactly 25 ASCII bytes: the TAI64N external representation of the server’s current time, formatted per [TAI64]:

’@’ hex16(seconds-since-TAI-epoch) hex8(nanoseconds)

where hex16 and hex8 denote 16 and 8 lowercase hexadecimal digits respectively. The leading @ is literal.

The following headers MUST be present:

Header	Value
Content-Type	application/tai64n
Content-Length	25
Cache-Control	no-store
TAI-Leap-Seconds	sf-integer, e.g. 37

Table 1

TAI-Leap-Seconds carries the offset, in seconds, between TAI and UTC at the moment of the response, as a Structured Field Integer [RFC9651]. leap-seconds emitted MUST be a non-negative integer in [0, 2^32−1]; sf-integer supports values up to ±10^15 but values outside

[0, 2^32−1] cannot be encoded in leap-u32be. A verifier that receives a TAI-Leap-Seconds value outside this range MUST treat the response as unsigned. Its value is included in the signed payload as leap-u32be (see Section 6.1). A server MUST emit it on every successful response. It is a singleton field (Section 5.5 of [RFC9110]) and MUST NOT appear more than once in a response. A recipient that encounters more than one instance MUST treat the field as absent per Section 4.2 of [RFC9651]; the recipient then has no server-supplied TAI-to-UTC offset for this response and MUST treat the response as unsigned regardless of whether TAI-Signature is present.

5.4. Nonce Echo

A client MAY send a request header TAI-Nonce carrying an opaque byte string encoded as a Structured Field Binary [RFC9651]. If present on a GET request, the server MUST include a TAI-Nonce response field whose sf-binary value encodes the same octet sequence as the request field. TAI-Nonce is a singleton field (Section 5.5 of [RFC9110]) and MUST NOT appear more than once in a request or response. A recipient that encounters more than one instance MUST treat the field as absent per Section 4.2 of [RFC9651]. A server that treats a request TAI-Nonce as absent for this reason MUST NOT sign the response. A server MAY instead respond with 400 Bad Request when a request contains duplicate TAI-Nonce fields. A client that treats a response TAI-Nonce as absent MUST treat the response as unsigned regardless of whether TAI-Signature is present.

A server MUST NOT interpret the nonce. If a server receives a TAI-Nonce that is a malformed sf-binary it MUST treat the field as absent and MUST NOT sign the response. The decoded sf-binary value MUST contain at least 7 octets and MUST NOT exceed 129 octets. When serialized as sf-binary, these correspond exactly to field values of 14 octets (: + 12 base64 + padding + :) and 174 octets (: + 172 base64 + :) respectively; implementations MAY reject on wire length before decoding as an optimisation, but the normative check is on decoded length.

The lower bound of 7 decoded octets provides sufficient client-supplied entropy for the replay defence in this section. The upper bound of 129 decoded octets caps the nonce’s contribution to overall response size; combined with the 25-byte body and the other protocol-defined response headers, the protocol-mandated fields contribute at most approximately 530 bytes in total, excluding transport framing and intermediary-added headers. Implementations that require larger nonces for compatibility with a higher-entropy convention may negotiate a future extension; this version of the protocol holds the bounds fixed for interoperability.

A server that receives a TAI-Nonce outside this range MUST NOT sign the response. A zero-length or syntactically empty field value is below the minimum and MUST be treated as absent. Clients SHOULD choose nonces that are unpredictable to a network observer when freshness matters; 16 random bytes encoded as an sf-binary value are sufficient for typical use.

When a client included a TAI-Nonce in the request, the trust level of the response is determined as follows. If the response carries no TAI-Nonce field, the response is Plain (level 0) regardless of whether a signature is present; a conformant server only signs responses that echo a nonce (see Section 6), so any signature in such a response MUST be ignored. If the response carries a TAI-Nonce whose decoded octets do not match the request nonce, the response is Inconsistent (level -1) and MUST be rejected regardless of signature state. If the response carries a TAI-Nonce whose decoded octets match the request nonce, the trust level depends on the key/signature state as defined in Section 8. The key/signature state is Unresolvable when TAI-Key-Selector is malformed or present but does not resolve in DNS. These cases are summarised in the following table.

Response nonce	Matches	Key/Sig	Outcome	Level
Absent	N/A	Any	Plain	0
Present	No	Any	Inconsistent	-1
Present	Yes	Absent	Unique	1
Present	Yes	Malformed	Unique	1
Present	Yes	Unresolvable	Unique	1
Present	Yes	Invalid	Inconsistent	-1
Present	Yes	Valid	Signed	2

Figure 1: Client trust level by response nonce and key/signature state.

A network attacker who strips the TAI-Nonce from the outgoing request causes the server to respond at level 0. A client that requires level 1 or above treats this as a denial of service; a client that accepts level 0 receives no freshness guarantee.

6. Signing

A Taistamp service MAY be configured with an Ed25519 [RFC8032] signing key and a selector. When both are configured, the server signs every response that carries a TAI-Nonce.

A signed successful response carries two additional headers:

Header	Value
TAI-Key-Selector	the selector, as an sf-token
TAI-Signature	the Ed25519 signature, as an sf-binary

Table 2

The selector MUST match the following ABNF production [RFC5234]. The grammar is constrained to the subset of DNS LDH labels ([RFC1035] Section 2.3.1) that are also valid sf-tokens ([RFC9651]); the leading-ALPHA requirement follows from sf-token, not from DNS alone, because sf-token requires its first character to be ALPHA or * ([RFC9651] Section 4.1.7), excluding digit-leading labels that DNS would otherwise permit:

```
selector = ALPHA [ *61( ALPHA / DIGIT / "-" )
                   ( ALPHA / DIGIT ) ]
```

TAI-Key-Selector is a singleton field (Section 5.5 of [RFC9110]) and MUST NOT appear more than once in a response. A recipient that encounters more than one instance MUST treat the field as absent per Section 4.2 of [RFC9651]; the response MUST then be treated as unsigned.

TAI-Signature is a singleton field (Section 5.5 of [RFC9110]) and MUST NOT appear more than once in a response. A recipient that encounters more than one instance MUST treat the field as absent per Section 4.2 of [RFC9651]; the response MUST then be treated as unsigned.

A response that carries TAI-Signature without TAI-Key-Selector MUST be treated as unsigned; the absent selector yields key/signature state Absent, as if no signature were present.

6.1. Framed Payload

The signed payload is the concatenation, in order, of the following fields:

```
domain-tag    = "taistamp-v1" %x00
label-bytes   = 25 octets ; the response body
leap-u32be    = 4 octets  ; TAI-Leap-Seconds,
              ;   big-endian unsigned
selector-len  = 1 octet   ; length of selector-bytes, 1..63
selector-bytes = 1..63 octets ; the selector, ASCII
nonce-bytes   = octets    ; decoded sf-binary octets
              ;   of the TAI-Nonce field value

payload = domain-tag label-bytes leap-u32be
          selector-len selector-bytes nonce-bytes
```

nonce-bytes is the octet sequence obtained by parsing the TAI-Nonce field value as an sf-binary item per [RFC9651] and decoding the base64 value. The textual representation of the field is not part of the signed input; only the decoded bytes are signed.

The trailing NUL on domain-tag is significant: it prevents extension of the tag into adjacent bytes by an attacker who controls the protocol version string.

The selector is length-prefixed by a single byte. A downgrade attacker who rewrites TAI-Key-Selector on the wire cannot match an existing signature without also matching its length, as the signature is bound to the exact payload including the length byte; forging a valid signature for a different selector length is computationally infeasible under the existential unforgeability of Ed25519.

The nonce occupies the tail of the payload. Its length is therefore self-delimiting: a verifier computes nonce-bytes from the decoded TAI-Nonce sf-binary value and appends it as the final payload component, with no explicit length field required.

6.2. When to Sign

A server MUST sign a response if and only if all of the following hold:

* The server is configured with a key and selector.
* The request carried a TAI-Nonce header.
* The response is a 200 to GET.

A request without a nonce MAY receive an unsigned response even when the server is configured to sign. This allows lightweight unsigned probes to coexist with authenticated reads, though the response itself MUST remain Cache-Control: no-store because the label is time-sensitive.

6.3. Intermediary Handling

An intermediary (proxy or gateway) that forwards a Taistamp request or response MUST forward the TAI-Nonce, TAI-Leap-Seconds, TAI-Key-Selector, and TAI-Signature fields intact and MUST NOT modify or remove them. Stripping or rewriting TAI-Nonce on a request prevents the server from signing the response. Stripping or rewriting any of the response fields prevents the client from verifying the signature.

These requirements apply to Taistamp-aware intermediaries. General-purpose proxies that do not recognise TAI-* fields may strip or modify them without violating their own specifications. Operators who require signed responses SHOULD avoid placing non-transparent intermediaries between client and server, or verify that any intermediaries are configured to forward these fields intact.

7. Key Publication

This section applies only to servers configured with a signing key and selector.

A server publishes its verification key as a DNS TXT record. The owner name is

<selector> "._taistamp." <host>

where <host> is the authority component of the URL at which /.well-known/taistamp is served, and <selector> is the value the server emits in TAI-Key-Selector.

The TXT record value is a single string, of length at most 255 octets, formatted as a tag-value list in the DKIM/DMARC style:

v=tai1; k=ed25519; p=<base64-of-32-raw-pubkey-bytes>

Tag	Value
v	Protocol version. tai1 for the framing in this document.
k	Key algorithm. ed25519 is the only algorithm currently defined.
p	Public key, standard base64 of the 32 raw Ed25519 public key bytes.

Table 3

Parsing is done as specified in Section 3.2 of [RFC6376]. A verifier MUST treat unknown tags as informational. A verifier MUST treat an unknown v or k as an unrecoverable failure for that selector and MUST NOT fall back silently.

7.1. Key Rotation

Key rotation follows a publish-then-switch-then-retire sequence.

Operators SHOULD choose a DNS TTL for the TXT record that balances query load against key rotation speed. A TTL between 3600 seconds (1 hour) and 86400 seconds (24 hours) is RECOMMENDED for stable keys. When an operator plans to rotate a key, they SHOULD lower the TTL of the existing TXT record well in advance of the rotation to ensure that verifier caches clear quickly once the new key is deployed.

Publish. The operator publishes a new TXT record under a new selector and waits at least one DNS TTL interval for propagation before reconfiguring the server.

Switch. The server is reconfigured to sign responses with the new key and selector.

Retire. The old TXT record SHOULD remain in DNS for at least one DNS TTL interval after the switch, to allow verifiers that cached the old record to complete any in-flight verification. The old TXT record MAY then be removed.

7.1.1. Verifier Key Caching

A verifier MAY cache the parsed public key for a selector. Verifiers MUST respect the DNS TTL returned by the resolver and MUST NOT use a cached entry beyond that TTL. Verifiers MUST NOT apply a hardcoded minimum or maximum cache lifetime that overrides the TTL set by the authoritative operator. If verification of a cached key fails, the verifier SHOULD re-fetch the TXT record once before treating the signature as invalid.

7.1.2. Key Revocation

If a signing key is compromised, the operator MUST remove its TXT record immediately, without a retirement period. Removing the TXT record is the only revocation mechanism defined by this protocol; no out-of-band revocation channel is specified.

Verifiers that have cached the public key will continue to accept signatures made with it until their cached entry’s DNS TTL expires. The exposure window is therefore bounded by the TTL remaining on the

cached entry at the moment of revocation. Operators who require rapid revocation SHOULD configure a short TTL on key records, accepting the trade-off of increased DNS query load.

8. Trust Levels

A Taistamp response is assigned one of four trust levels, combining the nonce echo result and the key/signature state.

Level 2 — Signed. The response TAI-Nonce matches the request nonce and the TAI-Signature verifies against the public key resolved from TAI-Key-Selector. The time value is both fresh and authenticated.

Level 1 — Unique. The response TAI-Nonce matches the request nonce, but the signature is absent, the TAI-Key-Selector is malformed, or the TAI-Key-Selector does not resolve in DNS. The time value is fresh but unauthenticated. A malformed or unresolvable selector indicates a configuration error or that the signing key has not yet propagated in DNS; this is a soft degradation, not a failure.

Level 0 — Plain. The response carries no echoed TAI-Nonce. The time value carries no freshness or authentication guarantee from this protocol.

Level -1 — Inconsistent. Either the response TAI-Nonce is present but does not match the request nonce, or the TAI-Key-Selector resolves but the TAI-Signature does not verify. The response MUST be rejected.

A client MUST NOT use a response at trust level -1. Whether a client accepts level 0 or level 1 when a higher level is expected is an application policy decision outside the scope of this document.

9. Verification

This section applies when the response carries both TAI-Key-Selector and TAI-Signature. A verifier reads the response and reconstructs the framed payload as defined in Section 6. The verifier MUST:

1. Confirm that the decoded sf-binary octets of the response TAI-Nonce match those of the request TAI-Nonce.
2. Confirm that the TAI-Key-Selector value is a well-formed selector matching the ABNF in Section 6.
3. Resolve the TXT record at <selector>._taistamp.<host> and parse the v, k, and p tags.
4. Verify the Ed25519 signature using the public key from p over the framed payload.

Each step produces a trust level outcome. If step 1 fails (nonce mismatch), the response is Inconsistent (level -1) and MUST be rejected. If step 2 fails (malformed selector) or step 3 yields no DNS record (unresolvable selector), signature verification stops and the response is Unique (level 1). If step 4 fails (signature does not verify), the response is Inconsistent (level -1) and MUST be rejected. If all four steps succeed, the response is Signed (level 2). Trust levels are defined in Section 8.

Verifiers MUST use the strict verification procedure described in Section 5.1.7 of [RFC8032].

10. Security Considerations

10.1. Threat Model

Taistamp authenticates a single time reading at the moment a client asks. It does not establish a long- running synchronised clock and does not bound network delay; a client is responsible for treating the returned label as a measurement subject to the round-trip time of its request. The trust level framework (see Section 8) expresses the combined freshness and authentication guarantees that a response carries.

10.2. Replay

A response is bound to its request only via the TAI-Nonce echo and its inclusion in the signed payload. A client that omits the nonce receives a Plain (level 0) response with no freshness guarantee. A client that reuses a nonce across requests gains no replay protection even at level 1 or 2, since the nonce no longer uniquely identifies the exchange. Clients SHOULD generate a fresh, unpredictable nonce for each request.

Replay protection is scoped to the client’s own nonce uniqueness enforcement. The protocol does not prevent a recorded request/ response pair from being presented to a second application that does not track nonce lifecycle. Clients SHOULD treat each Unique (level 1) or Signed (level 2) response as valid for a single use and SHOULD NOT accept a response whose nonce was issued by a previous request.

10.3. Domain Separation

The taistamp-v1%x00 prefix prevents an Ed25519 key configured for Taistamp from being induced to sign a message valid under another protocol that happens to embed the same suffix. Operators MUST NOT reuse a Taistamp signing key for any other purpose.

10.4. DNS Trust

A Signed (level 2) response reduces to trusting the DNS lookup of the TXT record at <selector>._taistamp.<host>. A verifier that does not validate DNS responses (for instance via DNSSEC or a trusted resolver path) is vulnerable to a DNS attacker who can substitute a public key and then sign responses with the corresponding private key, producing a forged time value that verifies successfully. Unlike DKIM, where key substitution causes verification failures, here it enables complete spoofing. Without authenticated DNS, a response that appears Signed (level 2) provides no stronger guarantee than Unique (level 1). A Unique (level 1) response does not involve DNS key resolution and is therefore not subject to this attack, though it provides no key authentication.

Operators SHOULD publish under a DNSSEC-signed zone. Verifiers SHOULD validate DNSSEC signatures or use a resolver path that provides equivalent guarantees.

10.5. Clock Manipulation

A signature attests that the server emitted a given label; it does not attest that the server's clock is correct. An operator whose clock is drifting or deliberately set wrong will sign whatever label its clock produces. Clients that depend on Taistamp for high-stakes decisions SHOULD compare readings from multiple independently operated services.

10.6. Leap-Second Reporting

The TAI-Leap-Seconds header is included in the signed payload as leap-u32be; the signature attests that the server emitted this specific value, but not that it is correct. Signing it allows the verifier to reconstruct the server's own UTC interpretation of the timestamp, even though the verifier need not trust that value for UTC conversion purposes. An operator whose leap-second table is wrong, or who deliberately sets it wrong, will sign whatever count they emit. Clients who require UTC equivalence MUST source the TAI-to-UTC offset from a trusted table rather than from the server.

10.7. Key Revocation Window

When a TXT record is deleted to revoke a compromised key, verifiers that have cached the corresponding public key will continue to assign Signed (level 2) to responses made with that key until their cached entry's DNS TTL expires. An attacker in possession of the compromised private key can forge responses during this window; the forged signatures will verify against the cached key, assigning

Signed (level 2) when the correct assignment — given the revoked key — would be Unique (level 1) at best.

The exposure window is bounded by the DNS TTL remaining on the cached entry at the moment of revocation. Verifiers that honour the TTL limit required by Section 7.1.1 will naturally limit this window to at most one TTL interval. Operators who require a short revocation window SHOULD configure a low DNS TTL on key records.

10.8. Selector-Based DNS Amplification

A verifier that performs one DNS lookup per incoming response carrying a selector that is malformed or does not resolve (Unresolvable key/signature state, yielding Unique (level 1) per Section 8) amplifies query load onto the authoritative server publishing the _taistamp.<host> TXT records. Two cases arise: a cached key that fails verification triggers a re-fetch per Section 7.1.1; a previously unseen selector triggers a first-time lookup, which the per-name re-fetch limit does not bound. Verifiers MUST apply rate-limiting or exponential back-off on all DNS lookups for names under _taistamp.<host>, regardless of whether the lookup is a first fetch or a re-fetch.

10.9. Caching Intermediaries

Note: this section is non-normative.

Successful GET responses carry Cache-Control: no-store (see Section 5.3), which instructs intermediaries not to retain or serve the response from cache. However, CDN and proxy configurations sometimes override this directive, for instance via path-based cache rules that match /.well-known/ resources. A client that included a TAI-Nonce is protected: a cached response carries a different nonce, producing Inconsistent (level -1), which MUST be rejected. A client that did not include a TAI-Nonce receives a Plain (level 0) response with no indication that the time value is stale.

Operators deploying Taistamp behind a CDN or reverse proxy SHOULD explicitly configure the intermediary to pass GET responses to /.well-known/taistamp through without caching, and SHOULD verify this behaviour before relying on Signed (level 2) responses for time-sensitive decisions.

11. IANA Considerations

11.1. Well-Known URI

IANA is requested to register the following entry in the "Well-Known URIs" registry [RFC8615]:

Field	Value
URI Suffix	taistamp
Change Controller	Apptly Software Ltd
Reference	This document
Status	permanent
Related Information	none

Table 4

11.2. Media Type

IANA is requested to register the media type application/tai64n in the "Media Types" registry per [RFC6838]:

* Type name: application
* Subtype name: tai64n
* Required parameters: none
* Optional parameters: none
* Encoding considerations: 7-bit (ASCII)
* Security considerations: see Section 10
* Interoperability considerations: none
* Published specification: this document
* Applications that use this media type: time- authentication clients
* Fragment identifier considerations: none
* Restrictions on usage: none
* Provisional registration: no
* Author/Change controller: Apptly Software Ltd

11.3. Header Fields

IANA is requested to register the following entries in the "Hypertext Transfer Protocol (HTTP) Field Name Registry" [RFC9110]:

Field Name	Status	Structured Type	Reference
TAI-Leap-Seconds	permanent	Integer	this document
TAI-Nonce	permanent	Binary	this document
TAI-Key-Selector	permanent	Token	this document
TAI-Signature	permanent	Binary	this document

Table 5

12. References

12.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, DOI 10.17487/RFC1035, November 1987, <https://www.rfc-editor.org/info/rfc1035>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC6376] Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed., "DomainKeys Identified Mail (DKIM) Signatures", STD 76, RFC 6376, DOI 10.17487/RFC6376, September 2011, <https://www.rfc-editor.org/info/rfc6376>.

[RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <https://www.rfc-editor.org/info/rfc8032>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

[RFC9651] Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", RFC 9651, DOI 10.17487/RFC9651, September 2024, <https://www.rfc-editor.org/info/rfc9651>.

[TAI64] Bernstein, D. J., "TAI64, TAI64N, and TAI64NA", 1997, <https://cr.yp.to/libtai/tai64.html>.

12.2. Informative References

[RFC3161] Adams, C., Cain, P., Pinkas, D., and R. Zuccherato, "Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP)", RFC 3161, DOI 10.17487/RFC3161, August 2001, <https://www.rfc-editor.org/info/rfc3161>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/info/rfc4033>.

[RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <https://www.rfc-editor.org/info/rfc5905>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8915] Franke, D., Sibold, D., Teichel, K., Dansarie, M., and R. Sundblad, "Network Time Security for the Network Time Protocol", RFC 8915, DOI 10.17487/RFC8915, September 2020, <https://www.rfc-editor.org/info/rfc8915>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[Roughtime]
Ladd, W. and M. Dansarie, "Roughtime", 2026, <https://datatracker.ietf.org/doc/draft-ietf-ntp-roughtime/>. Aanchal Malhotra and Adam Langley authored early drafts of this document.

[TAICLOCK] Bernstein, D. J., "TAICLOCK", 1997, <https://cr.yp.to/proto/taiclock.txt>.

Authors' Addresses

Alejandro Mery
Apptly Software Ltd
Email: amery@apptly.co
URI: https://apptly.co

Kroly Gbriel Nagy
Apptly Software Ltd
Email: nagy.karoly@apptly.co
URI: https://apptly.co