Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 26 November 2026

E. Cisbani
D. Ribaudo
Intesi Group
G. Damiano
Entrust

25 May 2026

Distributed Ledger Time-Stamp
draft-intesigroup-dlts-13

Abstract

This document defines a standard to extend Time Stamp Tokens with Time Attestations recorded on Distributed Ledgers.

The aim is to provide long-term validity to Time Stamp Tokens, backward compatible with currently available software.

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 26 November 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

Attesting that a file existed prior to a specific point in time can be useful - for example - to:

* prove when an agreement was signed, if it is disputed

* validate a signature after a revocation occurred

* prove the ownership for copyright

* grant record integrity

A Time-Stamp Token (TST) provided by a Time-Stamp Authority (TSA) compliant with [RFC3161] can be based on an accurate time source linked to Coordinated Universal Time, and can be very precise - it can prove the existence also at the second or less. It is such a consolidated standard that - for example - the European Union legally enforced its usage by eIDAS Regulation [eIDAS], European Standards and Technical Specifications [ETSI.EN.319.422] [ETSI.TS.101.861].

In an in-deep appraisal of Time Stamping Schemes conducted in 2001 by Masashi Une [IMES], PKI TSA was evaluated as one of the most desirables in term of security against alteration of a time stamp.

The integrity of the timestamping process that is inevitably bound to the integrity of the TSA gave rise to other proposals like [ANSI.X9.95] and [ISO.IEC.18014-4].

Furthermore a TSA TST can be validated for a limited time - usually no longer than 20 years for technical reasons such as the TSA certificates expiration, or for economic reasons such as the cost of providing the validation service by TSA.

This situation brought about some solutions [ETSI.TS.102.778_4] aimed at mitigating the inconvenience by extending the validity of TSA timestamps.

Security of a Distributed Ledger (def. in Section 2) is based on hashes of data timestamped and widely published. Each timestamp includes the previous timestamp in its hash, forming a chain, with each additional timestamp reinforcing the ones before it.

The advantage of a Distributed Ledger Attestation (DLA) relies on the resilience of the distributed system and the overall design whose aim is the DL perpetual survival.

Based on a distributed trust scheme, a Distributed Ledger significantly increases security as already noted by Haber and Stornetta in 1991 [HaberStornetta].

In the case of a permissioned DL, security is provided by an authoritative network of trust [Hyperledger], [NIST.IR.8202], while in the case of a permissionless DL security is provided by the economic incentive for running full nodes [Nakamoto].

On the other hand, a DLA is not yet a standard solution. Furthermore, the bigger the network the less precise the DLA, because distributed nodes need time to reach consensus.

Since a DLA turns out to be a complementary element providing long-term validity to TST - the aim of this specification is to allow an extension of the Time-Stamp Token for Distributed Ledger Attestations (DLA).

2. Terms 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.

This document also refers to the following terms and definitions:

Public Key Infrastructure: As defined in [RFC5280]

Trusted Third Party: As defined in [RFC3161]

Time-Stamping Authority: As defined in [RFC3161]

Time-Stamp Token: As defined in [RFC3161]

Time-Stamping Unit: As defined in [RFC3628]

Distributed Ledger: Various definitions of blockchain and distributed ledger technology exist, and some of these stress different technical features. Given the nature and scope of this document and the lack of definitional consensus we chose to use the term as defined by UK Government Chief Scientific Adviser [UK-GCSA] "A distributed ledger is essentially an asset database that can be shared across a network of multiple sites, geographies or institutions. All participants within a network can have their own identical copy of the ledger. Any changes to the ledger are reflected in all copies in minutes, or in some cases, seconds. The assets can be financial, legal, physical or electronic. The security and accuracy of the assets stored in the ledger are maintained cryptographically through the use of ’keys’ and signatures to control who can do what within the shared ledger. Entries can also be updated by one, some or all of the participants, according to rules agreed by the network".

Merkle Tree: As defined in [Merkle], [CrosbyWallach] and Section 2.1 of [RFC6962]

Aggregation Server: A server providing the aggregation of digests to be timestamped in a Merkle Tree. Digests submitted for aggregation are added to a list periodically combined into a single Merkle Tree. Then the digest at the root of that tree is timestamped on a Distributed Ledger.

Distributed Ledger Attestation: A Distributed Ledger (Timestamping)

Attestation is a proof or a promise of timestamping in a precise Distributed Ledger.

Calendar: A calendar is simply a collection of Distributed Ledger Attestations.

Calendar Server: A server providing remote access to a collection of Distributed Ledger Attestations.

3. Symbols and abbreviated terms

DL: Distributed Ledger

DLA: Distributed Ledger Attestation

PKI: Public Key Infrastructure

TSA: Time-Stamping Authority

TST: Time-Stamp Token

TSU: Time-Stamping Unit

TTP: Trusted Third Party

4. DL Attestation

A Digital Ledger can be seen as an untrusted logger - serving a number of clients who wish to store their events in the log - kept honest by a number of auditors who will challenge the logger to prove its correct behaviour [CrosbyWallach].

A Merkle Tree data structure accomplishes this in a very efficient way by aggregating many requests and submitting periodically to the log only the root digest of the tree. This log is built as a hash chain (aka blockchain) of small blocks of data. Consequently, the entire chain can be shared and maintained by a large number of nodes, becoming a distributed system.

In a permissioned DL the number of nodes can be small enough to permit a quick synchronization and reach consensus concerning the state of the chain. In a permissionless DL the large number of nodes introduces a relevant delay in order to reach consensus.

In the case of Bitcoin, for example, consensus is reached statistically. Usually in an average elapsed time of one hour six new blocks are added to the chain. A block of data that was added before the last six blocks, is considered to be practically immutable. This is due to the high computational power that would be required to rewrite the chain.

As a result of this scenario the elapsed time - from the request of aggregation of a digest to the proof consolidated inside the DL, may amount to one hour or more.

This is why we distinguish between a *promise* of attestation and a *proof* of attestation. Generally, an Aggregation Server provides only a promise to timestamp the client’s digest in the DL. However, when the aggregation is completed and the Merkle Tree root hash recorded in a block within the chain, the promise has not yet been confirmed.

Only after reaching consensus on that block can attestation be considered as proof, and made available by the Calendar Server.

For the sake of simplicity, the Aggregation Server and the Calendar Server can be implemented as a unique instance. In this document we will generically refer to a Calendar Server indicating both services.

The DLA data structure is out of scope in this specification document. Any Calendar Server can define his application protocol and data structure. For this specification the DLA is considered as pure data.

5. DL Time-Stamp Objects

The ASN.1 structure of Promise type is as follows:

```
Promise ::= SEQUENCE {
    version          INTEGER,
    calendarFormat   UTF8String,
    dlPromise        DLPromise,
    signerIdentifier issuerAndSerialNumber,
    serialNumber     INTEGER }

DLPromise ::= OCTET STRING
```

The ASN.1 structure of Proof type is as follows:

```
Proof ::= SEQUENCE {
    version          INTEGER,
    calendarFormat   UTF8String,
    dlProof          DLProof,
    signerIdentifier issuerAndSerialNumber,
    serialNumber     INTEGER }

DLProof ::= OCTET STRING
```

The fields of Promise and Proof type have the following meanings:

* version is the syntax version number. It MUST always be 0. The usage is as described in Section 1.3 of [RFC5652]

* calendarFormat is the media type format of the DL attestation. It MUST be a registered application media type, in accordance with procedures laid out in [RFC6838] - for example, if you wanted to use the [OpenTimestamps] format, the calendarFormat value would be the string "application/vnd.opentimestamps.ots" (without quotes) that is the IANA registered Media Type [OTS]

* dlProof and dlPromise are the proof and promise obtained from a Calendar Server using as input value the value of the signature field of the SignerInfo structure inside the digital signature of the TimeStampToken, as described in Section 5.3 of [RFC5652]

* signerIdentifier is an IssuerAndSerialNumber type that identifies the TSU signing certificate as described in Section 10.2.4 of [RFC5652]

* serialNumber is an integer assigned by the TSA to each TimeStampToken as described in Section 2.4.2 of [RFC3161]

5.1. DL Time-Stamp Attributes

A set of proofs or a set of promises, generated by a Calendar Server, MAY be included in a TST, using an unsigned attribute of the per-signer information.

To grant backward compatibility with any currently available software the unsigned attribute MUST be compliant with the specifications defined in Section 5.3 of [RFC5652] for Attribute type.

Attributes including a set of promises and a set of proofs MUST be unsigned attributes; they MUST NOT be signed attributes, authenticated attributes, unauthenticated attributes, or unprotected attributes.

The new objects MUST have the following OIDs where id-ce identifies the root of standard extensions as described in [RFC5280].

The ASN.1 structure of attributes including a set of promises is as follows:

```
id-ce-dltsPromises OBJECT IDENTIFIER ::= { id-ce TBD1 }

Promises            SET OF Promise
```

The ASN.1 structure of attributes including a set of proofs is as follows:

```
id-ce-dltsProofs OBJECT IDENTIFIER ::= { id-ce TBD2 }

Proofs              SET OF Proof
```

All the proofs and promises that have been returned MUST refer to the same parent TimeStampToken issued at the time of the request.

Note that a TSA can return a set of proofs and promises for the same input value as it can use calendar servers operating on different Distributed Ledgers.

5.1.1. Response Status

The response status code in the TimeStampResp MUST be compliant with the specifications described in Section 2.4.2 of [RFC3161] and Section 5.2.3 of [RFC4210].

According to the TimeStamp policy, when the response contains only a subset of the expected proofs and promises, the status field SHOULD contain either the value one (grantedWithMods) or the value two (rejection).

5.2. DL Time-Stamp Extensions

Upgrade from a set of promises to a set of proofs MAY be done requesting a new TST including inside a non critical extension the set of promises previously obtained in an unsigned attribute.

When the TSA receives a request which has a non critical extension containing a set of promises, it MAY request the Calendar Server to get the corresponding proof for each of them, and MAY include the set of proofs in the TST response, using a non critical extension of the TSTInfo sequence.

To grant backward compatibility with any currently available software, request and response non critical extensions MUST be compliant with the specifications described in Section 2.4 of [RFC3161] and Section 4.2 of [RFC5280].

Conforming TSAs MUST mark these extensions as non-critical.

The ASN.1 structure of the proof request extension is as follows:

```
id-ce-dltsPromises OBJECT IDENTIFIER

Promises            SET OF Promise
```

The ASN.1 structure of the proof response extension is as follows:

```
id-ce-dltsProofs OBJECT IDENTIFIER

Proofs               SET OF Proof
```

The proofs returned in the extensions by the TSA MUST NOT refer to the TimeStampToken issued at the time of the request. Each Proof MUST contain the explicit reference to the pointing TimeStampToken with signerIdentifier (referring to the TSU certificate) and serialNumber (referring to the time stamp serial number), which have been received in the Promise structure of the proof request extension.

5.2.1. Response Status

The response status code in the TimeStampResp MUST be compliant with the specifications described in Section 2.4.2 of [RFC3161] and Section 5.2.3 of [RFC4210].

Compliant servers SHOULD also use the status field as follows:

* according to TimeStamp policy, when the response contains only a subset of the expected proofs, the status field SHOULD contain either the value one (grantedWithMods) or two (rejection)

* when in the response no proof can be returned, the status field SHOULD contain the value two (rejection)

* when all the received promises recognized by the Calendar Server are pending, the status field SHOULD contain the value three (waiting).

5.3. Use case

In order to clarify the use of the objects thus defined, the case of a subscription made by two actors at different times, using distinct time stamps, is illustrated below.

5.3.1. Promises

Since each signer applies a time stamp to his signature, the structure will be presented according to the following simplified scheme, in which each promise is inserted as an unsigned attribute of the time stamp to which it refers.

```
signature-1
    +--- timestampToken
                |--- signerIdentifier
                |--- serialNumber-1
                +--- id-ce-dltsPromises
                        +--- Promise
                                |--- version
                                |--- calendarFormat
                                |--- dlPromise
                                |--- signerIdentifier
                                +--- serialNumber-1
signature-2
    +--- timestampToken
                |--- signerIdentifier
                |--- serialNumber-2
                +--- id-ce-dltsPromises
                        +--- Promise
                                |--- version
                                |--- calendarFormat
                                |--- dlPromise
                                |--- signerIdentifier
                                +--- serialNumber-2
```

Figure 1: Figure 1

Although replicating the signerIdentifier and serialNumber information may seem redundant in the case of a single timestamp, it can never be ruled out that a second signature with a new timestamp will be added later.

When you also want to obtain the proof of attestation on the DL, the application will be able to collect the two promises and include them as extensions in a new timestamp request. The result would have the following structure:

```
              +--- timestampToken
                         |--- signerIdentifier
                         |--- serialNumber-3
                         +--- id-ce-dltsPromises
                                 +--- Proof
                                         |--- version
                                         |--- calendarFormat
                                         |--- dlPromise
                                         |--- signerIdentifier
                                         +--- serialNumber-1
                                 +--- Proof
                                         |--- version
                                         |--- calendarFormat
                                         |--- dlPromise
                                         |--- signerIdentifier
                                         +--- serialNumber-2
```

Figure 2: Figure 2

From this example it is evident that the signerIdentifier and serialNumber pair is necessary to uniquely identify the TimestampToken to which each Proof obtained refers.

It is up to the application to choose whether the new timestamp, containing the evidence, will be saved within the same document, containing the promises, or stored separately.

6. Security Considerations

Each security consideration described in Section 4 of [RFC3161] SHALL be evaluated designing TSA services that include DL Time-Stamp extensions.

When a TSA executes a request to a Calendar Server the use of a nonce is RECOMMENDED because using a nonce always allows the client to detect replays.

Safety and reliability of the DL proofs depends on the robustness of the hash algorithms and on the stability of the DL, i.e. how expensive or difficult it would be for an attacker to alter the DL.

7. IANA Considerations

This document does not require any action by IANA.

8. Normative References

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

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

[RFC3628] Pinkas, D., Pope, N., and J. Ross, "Policy Requirements for Time-Stamping Authorities (TSAs)", IETF, DOI 10.17487/RFC3628, RFC 3628, November 2003, <https://www.rfc-editor.org/info/rfc3628>.

[RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", IETF, DOI 10.17487/RFC4210, RFC 4210, September 2005, <https://www.rfc-editor.org/info/rfc4210>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", IETF, DOI 10.17487/RFC5280, RFC 5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", IETF, DOI 10.17487/RFC5652, BCP 70, RFC 5652, September 2009, <https://www.rfc-editor.org/info/rfc5652>.

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

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

9. Informative References

[ANSI.X9.95] American National Standards Institute (ANSI), "Trusted Time Stamp Management And Security", ANSI X9.95:2005, 2005, <https://webstore.ansi.org/Standards/ASCX9/ANSIX9952005>.

[CrosbyWallach]
Crosby, S. A. and D. S. Wallach, "Efficient Data Structures for Tamper-Evident Logging", August 2009, <http://static.usenix.org/event/sec09/tech/full_papers/crosby.pdf>.

[eIDAS] The European Parliament and the Council of the European Union, "Regulation (EU) No 910/2014 of the European Parliament and of the Council of 23 July 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing Directive 1999/93/EC", Regulation (EU) No 910/2014, 23 July 2014, <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32014R0910=EN>.

[ETSI.EN.319.422]
European Telecommunications Standards Institute, "Electronic Signatures and Infrastructures (ESI); Time-stamping protocol and time-stamp token profiles", ETSI EN 319.422, March 2016, <https://www.etsi.org/deliver/etsi_en/319400_319499/319422/01.01.01_60/en_319422v010101p.pdf>.

[ETSI.TS.101.861]
European Telecommunications Standards Institute, "Time stamping profile", ETSI TS 101.861, July 2011, <https://www.etsi.org/deliver/etsi_ts/101800_101899/101861/01.04.01_60/ts_101861v010401p.pdf>.

[ETSI.TS.102.778_4]
European Telecommunications Standards Institute, "Electronic Signatures and Infrastructures (ESI); PDF Advanced Electronic Signature Profiles; Part 4: PAdES Long Term - PAdES-LTV Profile", ETSI TS 102.778-4, December 2009, <https://www.etsi.org/deliver/etsi_ts/102700_102799/10277804/01.01.02_60/ts_10277804v010102p.pdf>.

[HaberStornetta]
Haber, S. and W. S. Stornetta, "How to Time-Stamp a Digital Document", 1991, <https://www.anf.es/pdf/Haber_Stornetta.pdf>.

[Hyperledger]
The Linux Foundation, "Hyperledger Architecture, Volume 1: Introduction to Hyperledger Business Blockchain Design Philosophy and Consensus", August 2017, <https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf>.

[IMES] Une, M., "The Security Evaluation of Time Stamping Schemes: The Present Situation and Studies (2001)", 2001, <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.23.7486>.

[ISO.IEC.18014-4]
ISO and IEC, "Information technology - Security techniques - Time-stamping services - Part 4: Traceability of time sources", ISO/IEC 18014-4, <https://www.iso.org/standard/59934.html>.

[Merkle] Merkle, R. C., "Secrecy, authentication, and public-key systems",
Standard Electronic Labs Technical Report No. 1979-1, June 1979, <http://www.merkle.com/papers/Thesis1979.pdf>.

[Nakamoto] Nakamoto, S., "Bitcoin: A Peer-to-Peer Electronic Cash System", 31 October 2008, <https://bitcoin.org/bitcoin.pdf>.

[NIST.IR.8202]
Dylan., Y., Peter., M., Nik., R., and S. Karen., "Blockchain technology overview", NIST IR 8202, DOI NIST.IR.8202, January 2018, <https://doi.org/NIST.IR.8202>.

[OpenTimestamps]
Todd, P., "OpenTimestamps: Scalable, Trust-Minimized, Distributed Timestamping with Bitcoin", 15 September 2016, <https://petertodd.org/2016/opentimestamps-announcement>.

[OTS] Cisbani, E., "IANA registered OpenTimestamps Media Type", 24 June 2021, <https://www.iana.org/assignments/media-types/application/vnd.opentimestamps.ots>.

[RFC6962] Laurie, B., Langley, A., and E. Kasper, "Certificate Transparency", IETF, DOI 10.17487/RFC6962, RFC 6962, June 2013, <https://www.rfc-editor.org/info/rfc6962>.

[UK-GCSA] UK Government Chief Scientific Adviser, "Distributed Ledger Technology: beyond block chain", January 2016, <https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/492972/gs-16-1-distributed-ledger-technology.pdf>.

Authors' Addresses

Emanuele Cisbani
Intesi Group
Via Torino 48
20123 Milano
Italy
Phone: +39 026 760 641
Email: ecisbani@intesigroup.com
URI: https://www.intesigroup.com

Daniele Ribaudo
Intesi Group
Via Torino 48
20123 Milano
Italy
Phone: +39 026 760 641
Email: dribaudo@intesigroup.com
URI: https://www.intesigroup.com

Giuseppe Damiano
Entrust
One Station Square
Cambridge
CB1 2GA
United Kingdom
Email: giuseppe.damiano@entrust.com
URI: https://www.entrust.com