Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 3 October 2025

G. Narea
Relaycorp

1 April 2025

DomainAuth Version 1
draft-narea-domainauth-00

Abstract

This document defines DomainAuth, a protocol to attribute digital signatures to domain names in such a way that verification can occur entirely offline without a prior distribution of public keys.

Each signature is distributed as part of a self-contained "signature bundle" that encapsulates a complete chain of trust comprising: (1) a DNSSEC chain from the root zone to a TXT record containing a public key or its digest, (2) an X.509 certificate chain from the key specified in the TXT record to the final signing key, and (3) the digital signature in the form of a CMS SignedData structure.

Finally, signatures can be attributed to the domain name itself (e.g. "example.com") or specific users (e.g. "alice" of "example.com").

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://docs.veraid.net/domainauth-spec/draft-narea-domainauth.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-narea-domainauth/.

Source for this draft and an issue tracker can be found at https://github.com/CheVeraId/domainauth-spec.

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/.

Copyright Notice

Table of Contents

1. Introduction 3
  1.1. Problem Statement 4
  1.2. Design Goals 5
  1.3. Conventions and Terminology 6
2. Protocol Overview 7
  2.1. Architecture 7
  2.2. Workflow Summary 8
    2.2.1. Organisation Setup 8
    2.2.2. Certificate Issuance 8
    2.2.3. Signature Bundle Production 9
    2.2.4. Signature Bundle Verification 9
  2.3. Trust Model 10
3. DNS Integration 10
  3.1. DNSSEC Configuration 11
  3.2. TXT Record 11
  3.3. TTL Override 12
  3.4. DNSSEC Chain Serialisation 12
4. X.509 Certificate Profile 13
  4.1. Organisation Certificate 13
  4.2. Member Certificate 14
  4.3. Intermediate Certificate 14
5. Member Id Bundle 14
6. CMS SignedData Structure 15
  6.1. Signature Types 16
    6.1.1. Member Signatures 16

6.1.2. Organisation Signatures 16
6.2. Signature Metadata 16
7. Signature Bundle 17
7.1. Verification Procedure 18
8. Cryptographic Algorithms 21
8.1. Digital Signature Algorithms 21
8.2. Hash Functions 22
9. Maximum Validity Period 22
10. Data Serialisation 23
11. Test Service 23
12. Implementation Status 23
12.1. VeraId 24
12.2. Letro 26
13. Security Considerations 27
13.1. DNSSEC Dependency 27
13.2. Phishing Attacks 28
13.3. Domain Ownership Changes 28
13.4. Offline Verification Limitations 28
13.5. Denial of Service 29
13.6. Key Management 29
13.7. Audit Trails 30
14. Privacy Considerations 31
15. IANA Considerations 31
15.1. DomainAuth Signature Algorithm Registry 31
15.2. DomainAuth Key Digest Type Registry 32
16. References 33
16.1. Normative References 33
16.2. Informative References 35
Appendix A. ASN.1 Schemas 36
Appendix B. OID Registry 38
Appendix C. Service Design Guidance 38
C.1. Service Security Considerations 39
C.2. Service User Experience Considerations 40
Appendix D. Implementation Guidance 40
Appendix E. Organisation Operation Guidance 41
E.1. TLD DNSSEC Considerations 42
E.2. Organisation Key Management 42
Appendix F. Acknowledgements 43
Author’s Address 43

1. Introduction

Public Key Infrastructures typically require continuous Internet connectivity for certificate validation or prior distribution of public keys, creating significant limitations for offline or intermittently connected environments. This document addresses the challenge of securely attributing content to domain names in scenarios where verification must occur entirely offline, without

reliance on real-time certificate status checking or pre-distributed trust anchors.

DomainAuth solves this verification challenge by creating self-contained "signature bundles" that encapsulate the complete trust chain required for validation. Each bundle comprises three cryptographically linked components: a DNSSEC chain extending from the DNS root to a domain's TXT record containing key material, an X.509 certificate chain from the domain to the signing entity, and a CMS SignedData structure containing the digital signature. This architecture leverages established standards whilst eliminating the need for continuous connectivity or prior trust establishment.

This specification defines the protocol components, data structures, and verification procedures that constitute the DomainAuth protocol. It covers the DNS integration mechanism, cryptographic requirements, certificate management practices, and signature verification processes.

1.1. Problem Statement

The protocol was initially designed and implemented to provide users of the offline messaging application Letro [LETRO] with identifiers that are customisable, user friendly, universally unique, and verifiable.

Letro is powered by the delay-tolerant network Awala [AWALA], which offers an end-to-end encrypted sneakernet to transport data between a region disconnected from the Internet and a location with access to the Internet. In the most extreme cases, this physical transport may take several weeks, during which users should be able to produce and verify digital signatures without relying on online services.

Attacks by powerful adversaries, such as nation-state actors, are part of the threat model, given that Awala and Letro explicitly target people disconnected from the Internet due to conflict or government-sponsored censorship.

Despite its origin in delay-tolerant networking, DomainAuth has broader applicability and can be useful when the Internet is available, such as the following use cases:

* Client authentication. A client could prove its identity to a server by sending a short-lived token signed with DomainAuth; this would be analogous to using a JSON Web Token [JWT], except that it can be verified without a prior distribution of public keys or remote operations. Alternatively, the client could sign each message sent to the server.

* Artefact signing. Documents, applications, libraries, and other files could be signed on behalf of a domain name, without vendor-specific gatekeeping mechanisms. This could unlock further use cases, such as enabling users to share original content whilst proving authenticity and integrity, instead of sharing URLs to resources that could be blocked at the network level.

* Peer-to-peer web hosting. A next-generation of websites could be hosted on a peer-to-peer network, with files reliably attributed to their respective domain names.

The present document is meant to provide the foundation for all the use cases above in a generic manner.

1.2. Design Goals

DomainAuth is designed with the following primary goals:

1. *Offline verification:* All signature bundles contain sufficient information to be independently verified without requiring external network queries.

2. *Decentralisation:* The protocol avoids the need for centralised authorities beyond the DNS hierarchy itself. Each domain owner has exclusive control over their domain and its associated members.

3. *User-friendly identifiers:* Identities are based on familiar, human-readable domain names and user names rather than cryptographically-derived values.

4. *Build upon well-established standards:* DNSSEC for securing DNS responses, X.509 for certificate management, and CMS for digital signatures.

5. *Minimal trust assumptions:* The protocol reduces trust dependencies by leveraging DNSSEC, limiting potential credential issuance attacks to DNS hierarchy operators (primarily IANA and TLD operators).

6. *Contextual binding:* Signatures are bound to specific "services", preventing their unauthorised use across different contexts.

1.3. Conventions and Terminology

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 following terms are used:

* *Organisation:* A domain name that participates in the DomainAuth protocol by configuring DNSSEC and publishing the necessary DomainAuth TXT record(s).

* *Member:* An entity that produces signatures on behalf of an organisation. Members are either:

  - *Users:* Identified by a unique user name within an organisation.

  - *Bots:* Acting on behalf of the organisation as a whole.

* *DomainAuth TXT Record:* A DNS TXT record at _domainauth.<domain> that contains the organisation’s public key information.

* *Organisation Certificate:* An X.509 certificate owned by an organisation that serves as the root of trust for all signatures produced on behalf of that organisation.

* *Member Certificate:* An X.509 certificate issued by the organisation certificate to a member.

* *Member Id Bundle:* A data structure containing a member certificate, its issuing organisation certificate, and the DNSSEC chain proving the authenticity of the organisation’s DomainAuth TXT record.

* *Signature Bundle:* A data structure containing a digital signature and chain of trust that enables offline verification. There are two types of signature bundles:

  - *Member Signature Bundle:* A signature bundle containing a signature produced by a member using their private key.

  - *Organisation Signature Bundle:* A signature bundle containing a signature produced directly by an organisation using its private key, with a required member attribution that assigns authorship of the content to a specific member.

* *DNSSEC Chain:* A sequence of DNS responses that allows a verifier to cryptographically validate the authenticity of a DNS record.

* *Service:* A protocol or system that employs DomainAuth signatures for a specific use case. Each service defines the context in which a signature is valid, and its own rules for signature creation and verification.

2. Protocol Overview

2.1. Architecture

DomainAuth is built on three foundational layers:

1. *DNS and DNSSEC Layer:*

    * Provides the domain name hierarchy and DNSSEC-based verification of domain ownership.

    * The DNSSEC chain connects the DNS root of trust to the organisation’s domain, enabling offline validation without prior key distribution.

    * The DomainAuth TXT record bridges DNSSEC and PKI by publishing the organisation’s public key information in a standardised, discoverable way.

2. *PKI Layer:*

    * Establishes a per-organisation PKI where each organisation issues certificates to its members.

    * The Organisation certificate functions as the domain-specific trust anchor that binds the organisation’s public key to its domain name.

    * Member certificates extend the organisation’s trust to specific members, containing identity information that enables them to produce signatures.

3. *Signature Layer:*

    * Enables members to produce digital signatures on behalf of their organisation.

    * Signature bundles package digital signatures with all verification material, enabling offline validation.

These layers interact differently depending on the signature type:

* In *member signatures*, the chain of trust flows from the DNSSEC chain to the organisation certificate, then to the member certificate, and finally to the signature, providing end-to-end cryptographic proof of authorship.

* In *organisation signatures*, the chain of trust flows from the DNSSEC chain directly to the organisation certificate and then to the signature, with member attribution provided as a claim rather than a cryptographic proof.

Furthermore, Member Id Bundles are a key architectural component that packages the complete chain of trust (DNSSEC chain, organisation certificate, and member certificate) into a single message, enabling members to produce verifiable signatures offline.

2.2. Workflow Summary

The DomainAuth protocol involves the following key workflows:

2.2.1. Organisation Setup

First, the organisation must have DNSSEC properly configured for its domain.

Then, the organisation must generate an asymmetric key pair and publish its public key information in a DomainAuth TXT record at _domainauth.<domain> as described in Section 3.2.

Multiple such records are allowed, which can be useful for key rotation or binding different keys to different services.

2.2.2. Certificate Issuance

The organisation must obtain an X.509 certificate for its public key, or reuse an existing certificate valid during the intended validity period.

When issuing a member certificate, the organisation must distribute it along with the organisation certificate. This can be done with a Member Id Bundle as specified in Section 5, which is desirable in services meant to be used offline as it also contains the DNSSEC chain.

2.2.3. Signature Bundle Production

A member would produce a signature bundle as follows:

1. Use their private key to produce a CMS SignedData structure, encapsulating the member’s certificate.

2. Obtain the DNSSEC chain from the DomainAuth TXT record. This chain could be provided by the organisation (e.g. in the form of a Member Id Bundle), retrieved from a cache, or resolved from DNS if online.

3. Construct a signature bundle with the CMS SignedData structure, the organisation certificate, and the DNSSEC chain.

Similarly, an organisation would produce a signature bundle as follows:

1. Use its private key to produce a CMS SignedData structure, without encapsulating the organisation’s certificate.

2. Obtain the DNSSEC chain from the DomainAuth TXT record, either from a cache or by resolving it from DNS if online.

3. Construct a signature bundle with the CMS SignedData structure, the organisation certificate, and the DNSSEC chain.

In both cases, the signer can choose to encapsulate the plaintext in the CMS SignedData structure or distribute it separately.

2.2.4. Signature Bundle Verification

The verification process involves validating the entire chain of trust as follows:

1. Verify the DNSSEC chain.

2. Verify the organisation certificate using the public key from the TXT record.

3. Determine the certificate of the signer of the CMS SignedData structure. If it is an organisation signature, use the organisation certificate. Otherwise, use the certificate of the member, which is encapsulated in the CMS SignedData structure.

4. Verify the CMS SignedData structure against the certificate of the signer.

5. Verify that the signature is valid for the intended service and time period.

Alternatively, the verifier can start with the digital signature, then verify the organisation certificate and finally the DNSSEC chain.

For more detailed information on the verification process, particularly regarding validity periods, see Section 7.1.

2.3. Trust Model

DomainAuth’s trust model is anchored in DNSSEC and offers key distinctions from traditional PKIs:

* *Domain-specific trust roots:* Each organisation can only issue certificates for itself and its members, enforcing a strict hierarchy where domain control is the only path to certificate issuance.

* *Self-contained verification:* The inclusion of complete DNSSEC chains and certificates in signature bundles enables fully offline verification without prior key distribution.

* *Credential lifecycle:* DomainAuth favours short-lived credentials over revocation mechanisms, reducing complexity for disconnected operation.

* *Trust inheritance:* By relying on DNSSEC, DomainAuth inherits both its security properties and limitations, with trust ultimately rooted in the DNS hierarchy, including the root zone and TLD operators.

3. DNS Integration

This document makes no distinction between different types of DNS zones, with the exception of the root zone which MUST NOT participate in DomainAuth. The root zone exclusion avoids representation challenges in user interfaces (where it would appear as a dot or empty string) and eliminates the need for implementations to handle this special case.

TLDs, apex domains, and subdomains are all treated equivalently. Any domain at any level in the DNS hierarchy, except the root zone, MAY implement DomainAuth. Each participating domain operates entirely independently from its parent zones, with no hierarchical relationship or inherited trust.

Throughout this document, the terms "domain" and "domain name" refer to any such zone without regard to its hierarchical position.

3.1. DNSSEC Configuration

Participating domains MUST have a complete DNSSEC chain of trust from the root zone to the DomainAuth TXT record.

Newly registered domains SHOULD wait at least the maximum validity period in Section 9 before enabling DomainAuth to prevent potential attacks using DNSSEC chains from previous domain owners.

3.2. TXT Record

Each organisation participating in the DomainAuth protocol MUST publish a TXT record at _domainauth.<domain> with the following fields separated by simple spaces:

1. *Version* (required): A positive integer denoting the version of the DomainAuth TXT record format, set to 0 (zero) for this version of the specification.

2. *Key Algorithm* (required): A positive integer denoting the key algorithm as registered in the "DomainAuth Signature Algorithm Registry" (Section 15.1). See also Section 8.1.

3. *Key Digest Type* (required): A positive integer denoting the key digest type used to identify the key, as registered in the "DomainAuth Key Digest Type Registry" (Section 15.2).

4. *Key Id* (required): The representation of the key identifier. For example, per Section 8.1, for an RSA key, this is the Base64-encoded digest of the public key.

5. *TTL Override* (required): A positive integer representing the number of seconds for the maximum validity period of signatures. This value MUST be at least 1 second and not exceed the limit specified in Section 9. Refer to Section 3.3 for more details.

6. *Service OID* (optional): An Object Identifier (in dotted decimal notation) binding the key to a specific service. If omitted, the key is valid for any service.

For example, the following TXT record specifies an RSA-2048 key identified by its SHA-512 digest with a TTL override of 24 hours (86,400 seconds) and no service binding:

```
_domainauth.example.com. IN TXT "0 1 3 dGhpcyBpcyBub3QgYSByZWFsIGtleSBkaWdlc3Q 86400"
```

Multiple TXT records MAY be published at the same zone to support different keys, key algorithms, or services.

3.3. TTL Override

The TTL override field in the DomainAuth TXT record enables verification of DNS records and DNSSEC signatures for longer periods than their respective specifications would allow, which is essential for delay-tolerant use cases where users may be offline for extended periods.

DNS records and DNSSEC signatures typically have TTL values that may be as short as a few minutes or hours. The TTL override mechanism allows the DNSSEC chain to remain verifiable for a significantly longer period, regardless of the TTL in such records. Refer to Section 9 for the maximum validity period.

During verification, the TTL override creates a restricted time window that extends backwards from the end of the requested verification period by the specified number of seconds. Verification will succeed if the DNSSEC signatures (RRSIGs) were valid at any point during this window, even if they would have expired according to their original validity periods.

For example, if a DNSSEC signature has a standard validity period of 3600 seconds (1 hour) but the DomainAuth TXT record specifies a TTL override of 604,800 seconds (7 days), a signature bundle can still be verified up to 7 days after creation, even when offline. If a verifier attempts to verify a signature bundle 5 days after it was created, the verification would succeed with the TTL override, whereas it would fail with only the 1-hour DNSSEC signature validity period.

3.4. DNSSEC Chain Serialisation

The serialised chain MUST be encoded as the ASN.1 DnssecChain structure below, where each OCTET STRING contains a complete DNS message as detailed in [DNS]:

```
DnssecChain ::= SET OF OCTET STRING
```

This chain MUST include all DNSSEC responses necessary to validate the _domainauth.<domain>/TXT record from the trust anchor. However, the root zone DS records SHOULD be omitted, since they will be ignored by verifiers as described in Section 7.1.

Implementations SHOULD optimise the serialisation to minimise redundancy and size whilst ensuring completeness for offline verification.

4. X.509 Certificate Profile

All X.509 certificates MUST comply with [X.509]. Additionally, each certificate MUST:

* Have a validity period of at least 1 second and not exceeding the limit specified in Section 9.

* Only use the algorithms specified in Section 8.

Additional requirements and recommendations apply to specific certificate types as described in the following sections.

4.1. Organisation Certificate

This is a certificate whose subject key is referenced by the DomainAuth TXT record. The following requirements and recommendations apply:

* Its Subject Distinguished Name MUST contain the Common Name attribute (OID 2.5.4.3) set to the organisation’s domain name with a trailing dot (e.g. example.com.).

* When the certificate is used to issue other certificates, the Basic Constraints extension from Section 4.2.1.9 of [X.509] MUST be present and marked as critical. Additionally, the CA flag MUST be enabled, and the Path Length Constraint SHOULD be set to the lowest possible value for the length of the intended certificate chains.

* When the certificate is used directly to sign CMS SignedData structures, the Basic Constraints extension MAY be absent. If present, it SHOULD have the CA flag disabled.

Whilst the organisation certificate is typically self-issued, it MAY be issued by another certificate authority. In such cases, the issuer of the organisation certificate and any other certificates in the certification path to the organisation certificate are not considered part of the DomainAuth protocol, and any such parent certificates SHOULD NOT be included in the Signature Bundle. Verification of the organisation certificate is performed solely using the public key referenced in the DomainAuth TXT record.

4.2. Member Certificate

* Its Subject Distinguished Name MUST contain the Common Name attribute (OID 2.5.4.3) set to the member's name in the case of users or the at sign (@) in the case of bots.

* The Basic Constraints extension from Section 4.2.1.9 of [X.509] MAY be absent. If present, it SHOULD have the CA flag disabled.

4.3. Intermediate Certificate

Organisations MAY issue intermediate certificates to delegate the responsibility of issuing member certificates and organisation signatures to other entities.

When an intermediate certificate is used, the Basic Constraints extension from Section 4.2.1.9 of [X.509] MUST be present and marked as critical. Additionally, the CA flag MUST be enabled, and the Path Length Constraint SHOULD be set to the lowest possible value for the length of the intended certificate chains.

Note that if an intermediate certificate is assigned a Common Name, it could also be used as a member certificate and it could therefore produce member signatures.

5. Member Id Bundle

The Member Id Bundle is a self-contained message that provides all the information needed for a member to produce verifiable signatures. It is serialised using ASN.1 with the following structure:

```
MemberIdBundle ::= SEQUENCE {
    version                  [0] INTEGER,
    dnssecChain              [1] DnssecChain,
    organisationCertificate  [2] Certificate,
    memberCertificate        [3] Certificate,
    intermediateCertificates [4] SET OF Certificate OPTIONAL
}
```

Where:

* version is the format version, set to 0 (zero) in this version of the specification.

* dnssecChain contains the serialised DNSSEC chain proving the authenticity of the organisation's DomainAuth TXT record.

* organisationCertificate is the organisation's X.509 certificate.

* memberCertificate is the X.509 certificate issued to the member by the organisation.

* intermediateCertificates is a set of X.509 certificates issued by the organisation to other entities that can sign member certificates. It SHOULD NOT include certificates extraneous to the chain between the organisation certificate and the member certificate.

The Member Id Bundle links the member to their organisation and provides all the cryptographic material needed to verify this relationship. It serves as a precursor to signature production and is typically distributed to members by their organisation’s certificate management system.

Member Id Bundles are not inherently confidential, as they contain only public information, but their integrity is critical for secure signature production.

6. CMS SignedData Structure

DomainAuth signatures use CMS SignedData structures as specified in Section 5 of [CMS], with additional requirements and recommendations:

* signerInfos field:

  - There MUST be exactly one SignerInfo.

  - The digest and signature algorithms MUST comply with Section 8.

  - The following signed attributes MUST be included:

    o Content Type attribute described in Section 11.1 of [CMS].

    o Message Digest attribute described in Section 11.2 of [CMS].

    o DomainAuth signature metadata attribute as outlined in Section 6.2.

  - The SignerInfo.sid field MUST use the issuerAndSerialNumber choice specified in Section 5.3 of [CMS].

* certificates field:

  - Any intermediate certificates between the organisation and the signer MUST be included.

- The organisation certificate SHOULD NOT be included, since it is already included in the Signature Bundle.

- Certificates outside the certification path between the organisation and the signer SHOULD NOT be included.

6.1. Signature Types

DomainAuth supports two distinct types of signatures, offering different levels of assurance and operational flexibility:

6.1.1. Member Signatures

Member signatures are produced by members (users or bots) using their own private key. They are suitable for scenarios requiring strong non-repudiation at the individual member level, or when members need to produce signatures whilst being offline for extended periods.

The member’s certificate MUST be included in the SignedData.certificates field.

6.1.2. Organisation Signatures

Organisation signatures are produced using the organisation’s private key or a delegated signing key. These signatures involve attributing the signature to a member and are suitable when member certificate management is impractical.

The SignerInfo structure MUST include the DomainAuth member attribution in its signed attributes, using the OID 1.3.6.1.4.1.58708.1.2 and the value defined in the ASN.1 structure below:

```
MemberAttribution ::= UTF8String
```

The member attribution value MUST conform to the restrictions defined in Section 13.2. For users, this is the user name; for bots, this is the at sign (@).

Member attribution is a claim made by the organisation, not cryptographically proven by the member. Verifiers SHOULD present this distinction clearly to end users.

6.2. Signature Metadata

Each SignedData structure includes metadata that binds the signature to a specific service and validity period. This metadata is included as a signed attribute in the SignerInfo structure.

The signature metadata attribute MUST use the OID 1.3.6.1.4.1.58708.1.0 and be encoded as the SignatureMetadata ASN.1 structure below:

```
SignatureMetadata ::= SEQUENCE {
    serviceOid      [0] OBJECT IDENTIFIER,
    validityPeriod  [1] DatePeriod
}

DatePeriod ::= SEQUENCE {
    start  [0] GeneralizedTime,
    end    [1] GeneralizedTime
}
```

Where:

* serviceOid is the Object Identifier of the service for which the signature is valid.

* validityPeriod specifies the time period during which the signature is considered valid. The start and end fields MUST be expressed in Greenwich Mean Time (GMT) and MUST include seconds. Therefore, both times will follow the format YYYYMMDDHHMMSSZ. Both the start and end times are inclusive, meaning the signature is valid at exactly the start time and remains valid until exactly the end time.

7. Signature Bundle

The Signature Bundle is the primary artefact of the DomainAuth protocol, containing a digital signature and all the information needed to verify it offline. It is serialised using ASN.1 with the following structure:

```
SignatureBundle ::= SEQUENCE {
    version                 [0] INTEGER,
    dnssecChain             [1] DnssecChain,
    organisationCertificate [2] Certificate,
    signature               [3] ContentInfo
}
```

Where:

* version is the format version, set to 0 (zero) in this version of the specification.

* dnssecChain contains the serialised DNSSEC chain proving the authenticity of the organisation’s DomainAuth TXT record.

* organisationCertificate is the organisation’s X.509 certificate.

* signature is a CMS ContentInfo containing the SignedData structure.

The specific contents of the signature field depend on whether it is a member signature or an organisation signature, as detailed in Section 6.

7.1. Verification Procedure

Implementations MUST verify the syntactic validity of the signature bundle against its ASN.1 schema and reject malformed values. Refer to Section 10 for more information on serialisation formats.

A fundamental aspect of the verification procedure is to establish that all components—the DNSSEC chain, X.509 certificate path and the signature itself—were simultaneously valid for at least one second within the specified verification period. This temporal intersection of validity periods ensures the cryptographic continuity of the trust chain at the time of verification.

Implementations MUST verify every syntactically-valid signature bundle as follows, and fail if any step fails:

1. *Establish the verification parameters.* The verifier MUST specify the following parameters:

    * Plaintext: The content to be verified if it is detached from the SignedData structure (i.e. the field SignedData.encapContentInfo.eContent is absent). This value MUST NOT be provided if the plaintext is encapsulated.

    * Service: The OID of the service for which the signature must be valid.

    * Validity period: The inclusive time range during which the signature bundle must be valid for at least one second (e.g. 1st January 1970 00:00:00 UTC to 31st January 1970 23:59:59 UTC). This period MAY be specified as a specific time (e.g. 1st January 1970 00:00:00 UTC), in which case it MUST be converted to a 1-second period where the start and end are the same as the specified time.

    The verifier MAY override the root zone DS record(s) only for testing purposes or to reflect updated IANA trust anchors before the underlying DNSSEC implementation updates its copy.

2. *Identify the relevant DomainAuth TXT record and determine the verification time window for the DNSSEC chain:*

    1. Extract the domain name from the Common Name attribute in the organisation certificate’s Distinguished Name.

    2. Extract all records in the RRSet for _domainauth.<domain>/TXT.

    3. Parse each TXT record rdata, per the rules in Section 3.2.

    4. Locate records matching the subject key specification from the organisation certificate (key algorithm and key id) and the service OID specified by the verifier (either matching exactly or with an absent service OID). If multiple matching records exist, use the one with the specific service OID; if none exists, use the wildcard record. If multiple records of the same type (specific or wildcard) match, verification MUST fail.

    5. Extract the TTL override value from the identified TXT record.

    6. Calculate a verification time window for the DNSSEC chain as follows:

        * End time: The end of the required verification period (as specified by the verifier).

        * Start time: The maximum (later) of:

            - The start of the required verification period (as specified by the verifier).

            - The end time minus the TTL override value in seconds.

3. *Verify the DNSSEC chain* from the root zone to the _domainauth.<domain>/TXT RRSet as described in [DNSSEC]. The chain is considered valid if all DNSSEC signatures in the chain were simultaneously valid (i.e., not expired and not in the future) for at least one second within the verification time window calculated in the previous step.

4. *Verify the X.509 certificate chain* from the organisation certificate to the signer’s certificate as specified in [X.509], using any additional certificates in the SignedData.certificates field as potential intermediate certificates when constructing the chain. Note that the chain will comprise a single certificate when the organisation itself is the signer.

   The certificate chain MUST overlap with the verification time window and the DNSSEC chain for at least one second, meaning there must be at least one second during which all certificates in the chain were simultaneously valid and this second falls within both the verification time window and the period when the DNSSEC chain was valid.

5. *Verify the CMS SignedData structure* as described in Section 5.6 of [CMS], using the signer’s certificate from the SignedData.certificates field or the organisation certificate if the signer is the organisation itself.

   The signature metadata attribute MUST be present in the signed attributes of the SignerInfo structure. Additionally:

   * The service OID MUST match that specified by the verifier.

   * The validity period MUST overlap with the verification time window, the X.509 certificate chain and the DNSSEC chain for at least one second, meaning there must be a minimum one-second period during which the signature was valid, all certificates were valid, and the DNSSEC chain was valid.

   If present, the member attribution attribute MUST be in the signed attributes of the SignerInfo structure, and its value MUST be a valid member name as specified in Section 13.2. If absent, the signer MUST be a member whose certificate meets the requirements specified in Section 4.2.

6. *Produce verification output:*

   * The organisation name without a trailing dot (e.g. example.com). This string MUST be represented using Unicode, converting from Punycode if necessary.

   * If the signer is a user, their name MUST be a Unicode string. The name MUST be taken from the signer certificate in the case of member signatures, or from the member attribution in the case of organisation signatures. If the signer is a bot, no name MUST be produced (not even an empty string).

* Whether the signature was produced by the member or the organisation.

Alternatively, the verification MAY start with the SignedData structure and end with the DNSSEC chain as described below, as long as the validity periods across all components still overlap for at least one second:

1. Establish the verification parameters.

2. Verify the CMS SignedData structure.

3. Verify the X.509 certificate chain.

4. Identify the relevant DomainAuth TXT record and determine the verification time window for the DNSSEC chain.

5. Verify the DNSSEC chain.

6. Produce verification output.

The verification process MUST be performed in full, without skipping any steps.

8. Cryptographic Algorithms

This section describes the cryptographic algorithms used in the X.509 certificates and CMS SignedData structures, but not to the DNSSEC chain.

Algorithms not explicitly allowed by this specification MUST be rejected.

Services MAY recommend specific algorithms within the set of algorithms allowed by this specification.

8.1. Digital Signature Algorithms

For ease of adoption and interoperability reasons, this specification only requires support for RSA Signature Scheme with Appendix - Probabilistic Signature Scheme (RSA-PSS), as detailed in Section 8.1 of [RFC8017]. The DomainAuth Signature Algorithm Registry, as specified in Section 15.1, MAY introduce support for additional signature algorithms and restrict the use of RSA-PSS (including its deprecation).

Implementations MUST support RSA-PSS in X.509 certificates as specified in [RFC4055] and CMS SignedData structures according to [RFC4056].

RSA keys with moduli less than 2048 bits MUST be rejected. RSA keys with modulus size of 2048 MUST be supported, and greater sizes SHOULD be supported.

Due to the size of RSA public keys, they MUST be identified in TXT records using a digest. The process for creating this identifier involves:

1. Taking the public key in its ASN.1 DER-serialised form as a Subject Public Key Info (defined in Section 4.1.2.7 of [X.509]).

2. Computing a digest of this serialised key using a hash function from the DomainAuth Key Digest Type Registry (Section 15.2).

3. Encoding the resulting digest using Base64 without padding ([RFC4648]).

This encoded digest is then used as the Key Id in the TXT record.

8.2. Hash Functions

This section pertains specifically to the hash functions used within digital signatures themselves (e.g., X.509 certificates, CMS SignedData structures), distinct from their use in key identification.

For ease of adoption and interoperability reasons, implementations MUST support SHA-256, SHA-384, and SHA-512 as established in [NIST.FIPS.180-4].

The DomainAuth Key Digest Type Registry, as described in Section 15.2, separately governs the use of hash functions for key identification.

9. Maximum Validity Period

Digital signatures MUST NOT have a validity period greater than 7,776,000 seconds (90 days). This limit applies to DNSSEC RRSIG records, X.509 certificates, and CMS SignedData structures (including the signature metadata).

Similarly, verifiers MUST NOT allow a validity period greater than this limit when verifying signatures over a time period.

Services SHOULD specify a maximum validity period shorter than the protocol-level limit where feasible. This approach improves security by limiting the window of vulnerability in case of key compromise or other security incidents.

Applications built on DomainAuth services MAY impose even shorter validity periods based on their specific security requirements and threat models.

10. Data Serialisation

All data structures in the DomainAuth protocol are defined using Abstract Syntax Notation One (ASN.1), as referenced in [ASN.1].

Implementations MUST support Distinguished Encoding Rules (DER) as presented in [ASN.1].

Services MAY require or recommend additional ASN.1 encoding rules.

11. Test Service

Service-agnostic implementations SHOULD use the test service OID 1.3.6.1.4.1.58708.1.1 for testing purposes. Service implementations MUST NOT use this service.

This service is not subject to additional requirements or recommendations.

12. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC7942], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

Note to RFC Editor: Please remove this section before publication.

12.1. VeraId

DomainAuth is the successor to the VeraId protocol described in [VERAID]. DomainAuth and VeraId are functionally identical, except for the following differences:

* DNS TXT record:
  - Name: DomainAuth uses _domainauth.example.com., whilst VeraId uses _veraid.example.com..
  - Value: DomainAuth requires the value to begin with the number 0, denoting the version of the DomainAuth TXT record format, followed by a space. This value does not have a version number in VeraId.
* VeraId does not explicitly support intermediate certificates, and its implementations do not support them. Consequently, the intermediateCertificates field in the Member Id Bundle is not present in VeraId.
* VeraId only allows ASN.1 DER serialisation.
* Cryptographic algorithms:
  - Signature algorithms: VeraId only supports RSA-PSS with modulus sizes of 2048, 3072, and 4096 bits. Support for EdDSA signatures was considered, but not implemented due to lack of support in the target Hardware Security Modules (HSMs), as documented in https://issuetracker.google.com/issues/232422224 (https://issuetracker.google.com/issues/232422224).
  - Hash functions: VeraId only supports SHA-256, SHA-384, and SHA-512.

  - Key identification: VeraId only supports key digests (equivalent to Key Digest Types 1-3 in DomainAuth) as a consequence of its support for RSA keys only, and does not support direct use of public keys (Key Digest Type 0) for key identification.

* VeraId does not offer protection against IDN homograph attacks (see Section 13.2).

* VeraId only disallows at-signs (@), tabs, and new lines in user names. Otherwise, user names are case-sensitive and may contain spaces in VeraId.

* VeraId unnecessarily requires X.509 certificates to have the Authority Key Identifier and Subject Key Identifier extensions.

VeraId is led by the author of this document, who intends to deprecate the VeraId specification in favour of DomainAuth and update the reference implementations to fully comply with this specification.

The following reference implementations of VeraId are available, all implemented by Relaycorp and undergoing independent security audits with reports expected by mid-April 2025:

1. *VeraId JavaScript Library*

    * URL: https://github.com/relaycorp/veraid-js (https://github.com/relaycorp/veraid-js)

    * Maturity: Used in production in the VeraId Authority application.

    * Coverage: The entire protocol as outlined in [VERAID].

    * Licensing: MIT licence.

2. *VeraId JVM Library*

    * URL: https://github.com/relaycorp/veraid-jvm (https://github.com/relaycorp/veraid-jvm)

    * Maturity: Used in the Android version of Letro [LETRO].

    * Coverage: The entire protocol except for Organisation Signature Bundles.

    * Licensing: Apache 2.0 licence.

3. *VeraId Authority*

    * URL: https://github.com/relaycorp/veraid-authority (https://github.com/relaycorp/veraid-authority)

    * Description: A multi-tenant, cloud-native application for managing organisation members and issuing Member Id Bundles.

    * Maturity: Used in production in Letro [LETRO].

    * Coverage: Uses the VeraId JavaScript Library to issue Member Id Bundles and Organisation Signature Bundles.

    * Licensing: Business Source License version 1.1.

To facilitate functional testing, VeraId implementations use a dedicated VeraId TXT record at lib-testing.veraid.net, bound to the test service (Section 11), whose organisation's private key is publicly available in the source code for the projects above.

12.2. Letro

[LETRO] is the only VeraId service as of this writing.

* Organisation: Relaycorp

* URLs:

    - https://github.com/relaycorp/letro-android (https://github.com/relaycorp/letro-android)

    - https://docs.relaycorp.tech/letro-server/ (https://docs.relaycorp.tech/letro-server/)

* Maturity: Experimental.

* Coverage: The implementation exercises the entire protocol specified in [VERAID], except for organisation signatures and bot members. It uses the VeraId JVM Library to issue member signatures on Android, and the VeraId Authority to issue Member Id Bundles under a variety of domain names operated by Relaycorp (e.g. applepie.rocks, cuppa.fans).

* Licensing: Freely distributable with acknowledgement (GNU GPLv3 and GNU AGPLv3 licences).

* Contact: https://relaycorp.tech/ (https://relaycorp.tech/)

* Last updated: 2024

13. Security Considerations

13.1. DNSSEC Dependency

DomainAuth's security model relies fundamentally on DNSSEC, which introduces specific security considerations:

1. *Trust Anchors:*
    * The DomainAuth protocol inherits trust from the DNSSEC root zone.
    * Compromise of the root KSK would undermine the entire system.
    * Implementations MUST securely manage and update DNSSEC trust anchors.
    * Whilst the protocol allows overriding root zone DS records, this MUST only be done for testing or to reflect updated IANA trust anchors. Using alternative DNS roots in production contradicts the unique DNS root principle outlined in [RFC2826].
2. *TLD Control:*
    * Many TLDs are controlled by governments or private entities.
    * A malicious TLD operator could theoretically issue fraudulent DNSSEC responses for domains under their zone.
    * Organisations SHOULD consider the governance of their TLD when assessing security, including reviewing the relevant DNSSEC Practice Statement.
3. *DNSSEC Implementation Vulnerabilities:*
    * Flaws in DNSSEC implementations could affect DomainAuth security.
    * Implementations SHOULD use well-tested, actively maintained DNSSEC libraries.
    * Security updates for DNSSEC components SHOULD be promptly applied.

Whilst these dependencies introduce potential vulnerabilities, the distributed nature of DNS provides significant security advantages compared to centralised PKI models, particularly for offline verification scenarios.

13.2. Phishing Attacks

To mitigate Internationalised Domain Name (IDN) homograph attacks, user interfaces SHOULD adopt the guidelines from Section 2.11.2 (Recommendations for User Agents) of [UTR36].

To mitigate phishing attacks leveraging user names, such names MUST be normalised to the PRECIS UsernameCaseMapped profile as specified in Section 3.3 of [RFC8265], but spaces (U+0020) and at signs (U+0040) MUST NOT be allowed.

13.3. Domain Ownership Changes

Organisations SHOULD delay implementing DomainAuth until at least the period specified in Section 9 has elapsed since the domain was registered or acquired. This prevents the DNSSEC chain from the previous owner from remaining valid.

Domain ownership changes represent a fundamental challenge to any domain-based authentication system. DomainAuth’s approach of using short-lived certificates and signatures helps mitigate these risks by limiting the time window during which historical signatures remain valid.

13.4. Offline Verification Limitations

Offline verification introduces specific security considerations:

* *Time Synchronisation.* Verifiers rely on accurate local clocks to determine if signatures and certificates are within their validity periods. Devices with incorrect time settings may incorrectly accept expired signatures or reject valid ones.

* *Replay Attacks.* Services SHOULD implement additional measures (e.g., nonces) for replay-sensitive operations.

* *Revocation Limitations.* Offline verification cannot check real-time revocation status, and therefore the protocol relies on short validity periods rather than revocation checking.

13.5. Denial of Service

Several aspects of the protocol present potential denial of service vectors:

1. *Resource-exhaustion attacks*: Attackers could submit malformed or excessively complex signature bundles requiring significant resources to parse and validate, particularly during DNSSEC chain and certificate validation.

2. *Large bundles*: The protocol doesn't inherently limit signature bundle size. Services SHOULD enforce maximum bundle sizes appropriate to their use case to prevent resource exhaustion.

3. *Certificate chain complexity*: Bundles with extremely long certificate chains may cause excessive processing time or memory consumption during validation.

13.6. Key Management

Proper key management is essential for the security of the DomainAuth protocol. The following requirements and recommendations apply:

* *Key Generation.* DomainAuth REQUIRES a Cryptographically-Secure Pseudorandom Number Generator (CSPRNG) compliant with [RFC4086]. Implementations SHOULD integrate an existing CSPRNG instead of creating their own.

* *Key Storage.* Private keys MUST be protected from unauthorised access. Organisation private keys SHOULD be stored with the highest level of protection available, preferably in Hardware Security Modules (HSMs). Member private keys SHOULD be protected with appropriate measures, such as operating system security mechanisms or hardware tokens.

* *Key Rotation.* Organisations SHOULD establish a regular schedule for rotating their keys. Given the nature of the protocol, DomainAuth TXT records for old keys MAY be removed as soon as the new key is deployed without affecting signatures produced with the old key.

* *Key Compromise.* In the event of a key compromise, immediate rotation is REQUIRED and the compromised key's TXT record MUST be removed as soon as possible.

13.7. Audit Trails

Organisations implementing DomainAuth SHOULD maintain comprehensive audit logs of key management operations and certificate lifecycle events to support security incident investigation and facilitate non-repudiation.

* *Key Event Logging:*
  - Organisations MUST log key generation, rotation, and retirement events.
  - Each log entry SHOULD include the key identifier, timestamp, operation type, and authorised operator identity.
  - Key material itself MUST NOT be included in logs.
* *Certificate Operations:*
  - Certificate issuance and renewal events MUST be logged.
  - Logs SHOULD include certificate subject, issuer, serial number, validity period, and authorising entity.
* *Member Management:*
  - Member registration, name changes, and removal operations MUST be logged.
  - Logs SHOULD include the affected member identifier and the authorising entity.
* *Signature Operations:*
  - Organisations SHOULD log signature creation events for organisation signatures.
  - Member signatures MAY be logged depending on service requirements.
  - Logs SHOULD include signer identifier, signature validity period, service OID, and a content reference or hash.
* *Log Security:*
  - Audit logs MUST be protected against unauthorised modification and access.

- Logs SHOULD be stored with integrity protection mechanisms.

- In high-risk environments, organisations SHOULD implement tamper-evident logging.

14. Privacy Considerations

DomainAuth is focused on authentication and integrity. Confidentiality, a key property of privacy, is outside the scope of this protocol.

The protocol’s PKI design, where each organisation manages its own certificate hierarchy, helps avoid unnecessary information disclosure. By allowing an organisation to serve as the certificate authority for its members, this approach minimises the data that would otherwise need to be shared with verifiers, thereby reducing the risk of user information leakage or enabling user enumeration attacks.

Signature bundles do not contain sensitive personal data beyond what is necessary for identity verification, following the principle of data minimisation. The protocol also supports the use of bot signatures that do not identify specific individuals.

15. IANA Considerations

This document requests IANA to create a new registry group called "DomainAuth Protocol Parameters".

15.1. DomainAuth Signature Algorithm Registry

IANA is requested to create a new registry titled "DomainAuth Signature Algorithm Registry" under the "DomainAuth Protocol Parameters" registry group.

The registry policy is Specification Required as set forth in [RFC8126].

Each entry in this registry includes the following fields:

* Value: A numeric identifier for the signature algorithm.

* Algorithm: Name of the signature algorithm.

* Reference: Reference to the document that defines the algorithm.

* Status: Either "Active", "Deprecated", or "Obsolete".

The initial entries in this registry are:

Value	Algorithm	Reference	Status
1	RSA-PSS with modulus 2048 bits	(This document)	Active
2	RSA-PSS with modulus 3072 bits	(This document)	Active
3	RSA-PSS with modulus 4096 bits	(This document)	Active

Table 1: Initial entries in the DomainAuth Signature Algorithm Registry

The designated expert(s) MUST ensure that the proposed algorithm is appropriate for digital signatures and that the specification adequately defines its parameters and security properties. The expert MUST consider whether the algorithm has undergone sufficient security analysis and whether its inclusion would promote interoperability.

Algorithms MUST be marked as "Deprecated" if cryptographic vulnerabilities are discovered that could significantly affect their security properties. Algorithms SHOULD be marked as "Obsolete" if they should no longer be used due to serious security vulnerabilities. Implementers SHOULD NOT use deprecated algorithms for new signatures and MUST NOT use obsolete algorithms.

15.2. DomainAuth Key Digest Type Registry

IANA is requested to create a new registry titled "DomainAuth Key Digest Type Registry" under the "DomainAuth Protocol Parameters" registry group.

The registry policy is Specification Required as set forth in [RFC8126].

Each entry in this registry includes the following fields:

* Value: A numeric identifier for the key digest type.

* Hash Function: If applicable, the name of the hash function used to derive the key identifier from the public key.

* Reference: Reference to the document that defines the key digest type.

* Status: Either "Active", "Deprecated", or "Obsolete".

The initial entries in this registry are:

Value	Hash Function	Reference	Status
0	None	(This document)	Active
1	SHA-256	(This document)	Active
2	SHA-384	(This document)	Active
3	SHA-512	(This document)	Active

Table 2: Initial entries in the DomainAuth Key Digest Type Registry

Key Digest Type 0 (None) is reserved for future cryptographic algorithms where the public key itself is compact enough to be included directly in the Key Id field of the DomainAuth TXT record (Section 3.2).

The designated expert(s) MUST ensure that the proposed key digest type is cryptographically suitable for identifying public keys, and that the specification adequately defines its parameters and security properties.

Key digest types MUST be marked as "Deprecated" if cryptographic vulnerabilities are discovered that could significantly affect their security properties. Key digest types SHOULD be marked as "Obsolete" if they should no longer be used due to serious security vulnerabilities. Implementers SHOULD NOT use deprecated key digest types for new signatures and MUST NOT use obsolete key digest types.

Note to RFC Editor: Please replace all instances of (This document) with the RFC number of this document upon publication.

16. References

16.1. Normative References

[ASN.1] International Telecommunications Union, "Information Technology — ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, 1994.

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

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

[DNSSEC] Hoffman, P., "DNS Security Extensions (DNSSEC)", BCP 237, RFC 9364, DOI 10.17487/RFC9364, February 2023, <https://www.rfc-editor.org/rfc/rfc9364>.

[NIST.FIPS.180-4] Dang, Q. H. and NIST, "Secure Hash Standard", NIST Federal Information Processing Standards Publications 180-4, DOI 10.6028/NIST.FIPS.180-4, July 2015, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

[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/rfc/rfc2119>.

[RFC4055] Schaad, J., Kaliski, B., and R. Housley, "Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 4055, DOI 10.17487/RFC4055, June 2005, <https://www.rfc-editor.org/rfc/rfc4055>.

[RFC4056] Schaad, J., "Use of the RSASSA-PSS Signature Algorithm in Cryptographic Message Syntax (CMS)", RFC 4056, DOI 10.17487/RFC4056, June 2005, <https://www.rfc-editor.org/rfc/rfc4056>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/rfc/rfc4086>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/rfc/rfc7942>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[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/rfc/rfc8174>.

[RFC8265] Saint-Andre, P. and A. Melnikov, "Preparation, Enforcement, and Comparison of Internationalized Strings Representing Usernames and Passwords", RFC 8265, DOI 10.17487/RFC8265, October 2017, <https://www.rfc-editor.org/rfc/rfc8265>.

[UTR36] Davis, M. and M. Suignard, "UTR #36: Unicode Security Considerations", 19 September 2014, <https://www.unicode.org/reports/tr36/tr36-15.html>.

[X.509] 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", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

16.2. Informative References

[AWALA] Narea, G., "Awala", 2019, <https://specs.awala.network/>.

[JWT] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/rfc/rfc7519>.

[LETRO] Narea, G., "Letro", 2023, <https://letro.app/en/>.

[RFC2826] IAB, "IAB Technical Comment on the Unique DNS Root", RFC 2826, DOI 10.17487/RFC2826, May 2000, <https://www.rfc-editor.org/rfc/rfc2826>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/rfc/rfc8017>.

[VERAID] Narea, G., "VeraId V1 Specification", 2025, <https://veraid.net/spec/>.

Appendix A. ASN.1 Schemas

The following ASN.1 schemas define the data structures used in the DomainAuth protocol:

```
-- Top-level schemas for DomainAuth components

-- DNSSEC chain is a set of DNS messages
DnssecChain ::= SET OF OCTET STRING

-- Member Id Bundle
MemberIdBundle ::= SEQUENCE {
    version                 [0] INTEGER,
    dnssecChain             [1] DnssecChain,
    organisationCertificate [2] Certificate,
    memberCertificate       [3] Certificate,
    intermediateCertificates [4] SET OF Certificate OPTIONAL
}

-- Signature Bundle
SignatureBundle ::= SEQUENCE {
    version                 [0] INTEGER,
    dnssecChain             [1] DnssecChain,
    organisationCertificate [2] Certificate,
    signature               [3] ContentInfo
}

-- Signature metadata (included as a signed attribute)
SignatureMetadata ::= SEQUENCE {
    serviceOid      [0] OBJECT IDENTIFIER,
    validityPeriod  [1] DatePeriod
}

-- Date period structure
DatePeriod ::= SEQUENCE {
    start  [0] GeneralizedTime,
    end    [1] GeneralizedTime
}

-- Member attribution (included as a signed attribute in organisation signatures)
MemberAttribution ::= UTF8String
```

All DomainAuth data structures MUST be encoded using ASN.1 as specified in Section 10.

The ASN.1 structures reference standard types from other specifications:

* Certificate is defined in [X.509].

* ContentInfo is defined in [CMS].

All implementations MUST strictly adhere to these schemas.

Appendix B. OID Registry

The following Object Identifiers (OIDs) are defined for use in the DomainAuth protocol, under the 1.3.6.1.4.1.58708.1 (iso.org.dod.internet.private.enterprise.relaycorp.domainauth) arc:

* 1.3.6.1.4.1.58708.1.0: Signature Metadata Attribute.

* 1.3.6.1.4.1.58708.1.1: Test Service.

* 1.3.6.1.4.1.58708.1.2: Member Attribution Attribute.

Appendix C. Service Design Guidance

This non-normative section summarises key considerations for system designers creating services that utilise DomainAuth, drawing upon requirements and recommendations specified elsewhere in this document.

Service designers are required to obtain a unique OID for their service, distinct from the DomainAuth OID arc (see Appendix B), as this OID is embedded within the signature metadata (Section 6.2) and used during verification (Section 7.1). Note that organisations have the option to bind specific keys to a service OID via the DomainAuth TXT record (Section 3.2), which verifiers are required to account for when matching signatures to keys.

Consideration should be given to incorporating version information either within the OID structure or the service-specific plaintext data being signed, to facilitate future upgrades to the plaintext format.

Given the reliance on local clocks for offline verification (Section 13.4), services may find it beneficial to establish a minimum signature validity duration (e.g., several minutes) to accommodate potential clock skew. Remember that the overall verification requires temporal overlap between the DNSSEC chain’s validity window (influenced by the TTL override Section 3.3), the certificate chain’s validity (Section 4), and the signature metadata’s validity period (Section 7.1).

Whilst the protocol mandates ASN.1 DER encoding (Section 10), services have the option to specify alternative ASN.1 encoding rules, but service-level implementations bear the responsibility for any necessary conversions if the underlying DomainAuth library does not support the alternative rules (Section 10).

Services need to determine which signature types (Member, Organisation, or both) are appropriate for their use case (Section 6.1), and whether plaintext will be encapsulated or detached (Section 2.2.3).

C.1. Service Security Considerations

The validity period for signatures, specified in the signature metadata (Section 6.2), is required to adhere to the maximum limit defined in Section 9. It is recommended that services define their own maximum validity period, as short as possible for their use case, to enhance security (Section 9).

DomainAuth specifies mandatory-to-implement cryptographic algorithms (Section 8). Services have the option to recommend specific algorithms from the allowed set but cannot mandate unsupported ones.

Service implementations must not override root zone DS records except for testing or to reflect updated IANA trust anchors. Services using DomainAuth should clearly distinguish test environments from production.

Services designers should also consider the following potential attack vectors:

* *Replay attacks:* DomainAuth provides authenticity and integrity but does not inherently mitigate replay attacks. Services may need to implement additional measures, such as nonces, depending on the application’s sensitivity to replays (Section 13.4).

* *Phishing attacks:* Service logic built upon the verified identity ought to correctly handle the required Unicode representation of domain names, and the specific normalisation and character restrictions applicable to user names (Section 13.2, Section 7.1). Adhering to guidelines like those in [UTR36] for displaying Internationalised Domain Names can help mitigate homograph attacks.

* *Denial of service:* Service designers should establish limits on signature bundle size to mitigate resource exhaustion attacks (Section 13.5).

Services may define their own requirements for logging signature events, which would complement the organisation-level audit trails recommended in Section 13.7.

C.2. Service User Experience Considerations

Presenting DomainAuth verification results clearly is important for users to understand the basis of trust. Since DomainAuth supports different signature types with distinct trust implications, user interfaces can play a crucial role in conveying this information accurately.

Designers should consider the following points when crafting the user experience for DomainAuth services:

* *Communicating signature origin.* Consider distinguishing between member signatures (cryptographically proven) and organisation signatures (where the organisation vouches for content with member attribution). For organisation signatures, the attribution may be presented as a claim by the organisation (e.g., "Vouched for by example.com, attributed to alice") rather than implying direct proof from the member. However, the distinction may or may not be relevant depending on the service.

* *Representing bot signatures.* When displaying signatures associated with bots, the interface has to attribute the signature to the organisation itself, omitting any specific user name (e.g. "Signed by example.com").

* *Displaying identifiers safely.*

  - Consider implementing IDN homograph attack mitigations, such as those suggested in [UTR36].

  - Display the user name exactly as provided by the verification process (which is based on normalised data per Section 13.2).

  - Avoid truncating user names or domain names to prevent ambiguity or misrepresentation.

  - Visually distinguish the user name part from the domain name part of an identifier (e.g., "alice of example.com", "alice@example.com").

Appendix D. Implementation Guidance

This non-normative section provides guidance for developers implementing DomainAuth libraries.

Implementations should ensure strict compliance with the ASN.1 schemas (Appendix A), serialisation formats (Section 10), and cryptographic algorithms (Section 8). Separating core protocol functionality from service-specific logic enables reuse across multiple services.

For testing purposes, implementers should consider establishing a dedicated DNSSEC-enabled test domain with a DomainAuth TXT record whose private key is publicly available, bound specifically to the test service OID (1.3.6.1.4.1.58708.1.1). This approach is secure when properly scoped: the domain is controlled by the implementation team, the key is bound only to the test service, and documentation clearly indicates these are for testing only. Serialised DNSSEC chains can also serve as test fixtures, allowing offline testing.

Implementations have to provide the ability to override root zone DS records, which is only intended for local testing environments or to reflect updated IANA trust anchors before the underlying DNSSEC implementation is updated.

The verification procedure should support both verification orders described in Section 7.1: either starting with the DNSSEC chain and ending with the signature, or vice versa. A critical aspect is validating that all components in the trust chain (DNSSEC records, certificates, and signatures) have validity periods that overlap for at least one second. Implementations should also handle multiple TXT records correctly, selecting the appropriate one based on key information and service OID.

Error handling should categorise failures based on verification steps (Section 7.1), with different diagnostic detail levels for debugging versus production. Performance optimisations may include caching verified DNSSEC chains, using efficient ASN.1 parsing libraries, ensuring thread safety, and optimising certificate validation operations.

Appendix E. Organisation Operation Guidance

This non-normative section provides guidance for organisations operating domains that participate in the DomainAuth protocol.

Before implementation, organisations should ensure their domain has been under their control for at least the maximum validity period (Section 9) and consider TLD governance (Section 13.1). DNSSEC must be properly configured, with processes established for key rollovers.

E.1. TLD DNSSEC Considerations

When selecting a domain for DomainAuth implementation, organisations should carefully assess the security practices of the TLD operator, as mentioned in Section 13.1. Factors to consider include:

* The TLD operator’s jurisdiction and potential vulnerability to government influence.
* The cryptographic algorithms used for DNSSEC by the TLD.
* The TLD’s DNSSEC Practice Statement.
* The TLD operator’s history of security incidents or vulnerabilities.

For security-critical applications, organisations may wish to select TLDs with strong security practices and jurisdictions with robust legal protections.

Similar considerations apply to the DNS hosting provider used for the domain, as they represent another potential attack vector. The security practices, DNSSEC support, access controls, and jurisdictional factors of the entity responsible for hosting the domain’s DNS records can significantly impact the overall security of a DomainAuth implementation.

E.2. Organisation Key Management

Key management practices should include secure generation (potentially using HSMs as suggested in Section 13.6), protected storage, and scheduled rotation. Organisations may publish multiple TXT records for different keys, algorithms, or services, which facilitates smooth key rotation. The TTL override value in TXT records significantly affects verification windows and should be set appropriately for the service’s offline requirements.

The organisation certificate serves as the trust anchor for all members, making its protection particularly important. For member management, establish clear processes for certificate issuance, naming conventions that comply with normalisation requirements (Section 13.2), secure distribution of Member Id Bundles, and handling departing members.

Operational security measures should include regular verification of DomainAuth TXT records, comprehensive logging (Section 13.7), certificate inventory maintenance, and monitoring for unauthorised DNS modifications. Incident response procedures should address key

compromise, DNS hijacking, and trust recovery, noting that the limited validity period of signatures helps contain security incident impact.

Due to the residual risks regarding offline verification (Section 13.4), organisations should implement additional safeguards:

* Keep validity periods as short as operationally feasible.

* Implement robust monitoring for potential compromise indicators.

* Maintain secure backup procedures for cryptographic materials.

* Have emergency procedures in place to respond to security incidents, including communication plans to notify relying parties.

Appendix F. Acknowledgements

The author is grateful to the Open Technology Fund for funding the implementation of VeraId, which heavily influenced the final specification of the VeraId protocol, and therefore DomainAuth as its successor.

The author would also like to thank the authors of [DNS], [DNSSEC], [X.509], [CMS], and [ASN.1], which underpin the present protocol.

Author’s Address

Gus Narea
Relaycorp
Email: gus@relaycorp.tech