Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 10 August 2026

M. Riechert
A. Delignat-Lavaud
Microsoft
H. Birkholz
Fraunhofer SIT
A. Chamayou
Microsoft

6 February 2026

x509 Decentralized Identifier
draft-birkholz-did-x509-02

Abstract

This document defines the did:x509 decentralized identifier method, which enables a direct, resolvable binding between X.509 certificate chains and compact issuer identifiers (DID string). In particular, the did:x509 identifier format in this documents comes with a CWT Claims definition. In general, this identifier is a compact and interoperable mechanism for certificate-based identification by combining a certificate fingerprint with optional policies for subject names, subject alternative names, extended key usage, and issuer information. It is especially useful for policy evaluation and reference in transparency services and similar systems requiring cryptographic binding to certificate material.

This Informational document is published as an Independent Submission to improve interoperability with Microsoft’s architecture. It is not a standard nor a product of the IETF.

Discussion Venues

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

Source for this draft and an issue tracker can be found at https://github.com/henkbirkholz/draft-birkholz-did-x509.

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

This document aims to define an interoperable and flexible decentralized identifier ([DID]) format for COSE messages that transport or refer to X.509 certificates using [RFC9360]. The did:x509 identifier format implements a direct, resolvable binding between a certificate chain and a compact issuer string. It can be used in a COSE Header CWT Claims map as defined in [RFC9597].

Including a certificate chain directly in configuration or in policy is often impractical. This is due to its size, and to the frequency at which some elements, particularly the leaf, are refreshed. Relying on a partial certificate chain (e.g., a root certificate and some intermediary certificates) is similarly unwieldy. While stable, the level of granularity afforded by a partial certificate chain may not be sufficient to distinguish several identities that are not equivalent for the purpose of policy.

Combining authority pinning with attribute assertions is a precise and stable way of capturing identities as a constrained set of certificates. Their representation as compact and durable identifier strings enables the formulation of readable policy (e.g. "request.issuer == 'did:x509...'"), for example in the context of transparency ledger registration.

1.1. Conventions and Definitions

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

In this document, CDDL ([RFC8610], [RFC9165]) is used to describe the data formats, and ABNF (defined in [RFC5234]) to describe identifiers.

The reader is assumed to be familiar with the vocabulary and concepts defined in [I-D.ietf-scitt-architecture].

Rego is a descriptive query language used to define policies in a precise and unambiguous way. This document uses Rego ([REGO]) to define the parsing and validation logic for did:x509 identifiers. The Rego code snippets provided in this document can be evaluated using any Rego v1 runtime, but there is no expectation that implementations use the Rego language.

Per [RFC8792], line breaks may be present in the figures of this document to stay within the line-length limits of this document's format.

2. Identifier Syntax

The did:x509 ABNF definition defined below uses the syntax defined in [RFC5234] and the corresponding definitions for ALPHA and DIGIT. [DID] contains the definitions for idchar and pct-encoded in Section 3.1.

```
idchar             = ALPHA / DIGIT / "." / "-" / "_" / pct-encoded
pct-encoded        = "%" HEXDIG HEXDIG

=============== NOTE: '\' line wrapping per RFC 8792 ================

did-x509           = "did:x509:" method-specific-id
method-specific-id = version ":" ca-fingerprint-alg ":" ca-\
               fingerprint 1*("::" predicate-name ":" predicate-value)
version            = 1*DIGIT
ca-fingerprint-alg = "sha256" / "sha384" / "sha512"
ca-fingerprint     = base64url
predicate-name     = 1*ALPHA
predicate-value    = *(1*idchar ":") 1*idchar
base64url          = 1*(ALPHA / DIGIT / "-" / "_")
```

Figure 1: ABNF Definition of Core did-x509 Syntax

Implementations of this specification MUST indicate a version value of 0.

ca-fingerprint-alg is one of sha256, sha384, or sha512. ca-fingerprint is chain[i].fingerprint[ca-fingerprint-alg] with i > 0, that is, either an intermediate or root CA certificate. predicate-name is a predicate name and predicate-value is a predicate-specific value. :: is used to separate multiple predicates from each other.

The following sections define the predicates and their predicate-specific syntax.

Validation of predicates is defined using policies written in the Rego language ([REGO]), rather than pseudo-code. This is to avoid ambiguity and to make it possible for a reader to evaluate the logic automatically.

The inputs to the resolution process are the DID string itself and the x509chain DID resolution option, which carries a comma-separated base64url-encoded X.509 certificate chain. To evaluate the reference

Rego code shown below, the DID and certificate chain have to be passed to a Rego runtime as a JSON document: {"did": "<DID>", "chain": <CertificateChain>}, where did is the DID string and chain is the parsed representation of the certificate chain derived from the x509chain resolution option.

Core Rego policy:

```
parse_did(did) :=
  [ca_fingerprint_alg, ca_fingerprint, predicates] if {
    prefix := "did:x509:0:"
    startswith(did, prefix) == true
    rest := trim_prefix(did, prefix)
    parts := split(rest, "::")
    [ca_fingerprint_alg, ca_fingerprint] := split(parts[0], ":")
    predicates_raw := array.slice(parts, 1, count(parts))
    predicates := [y |
        some i
        s := predicates_raw[i]
        j := indexof(s, ":")
        y := [substring(s, 0, j), substring(s, j+1, -1)]
    ]
}

valid if {
    [ca_fingerprint_alg,
     ca_fingerprint,
     predicates] := parse_did(input.did)
    ca := [c | some i; i != 0; c := input.chain[i]]
    ca[_].fingerprint[ca_fingerprint_alg] == ca_fingerprint
    valid_predicates := [i |
        some i
        [name, value] := predicates[i]
        validate_predicate(name, value)
    ]
    count(valid_predicates) == count(predicates)
}
```

Figure 2: Core Rego Validation Rule

The overall Rego policy is assembled by concatenating the core Rego policy with the Rego policy fragments in the following sections, each one defining a validate_predicate function.

2.1. Percent-encoding

Some of the predicates that are defined in subsequent sections require values to be percent-encoded. Percent-encoding is specified in Section 2.1 of [RFC3986]. All characters that are not in the allowed set defined below must be percent-encoded:

```
allowed = ALPHA / DIGIT / "-" / "." / "_"
```

Figure 3: ABNF Definition of Characters That Do Not Need to Be Percent-Encoded

Note that most libraries implement percent-encoding in the context of URLs and do NOT encode ~ (%7E).

2.2. "subject" predicate

```
predicate-name  = "subject"
predicate-value = key ":" value *(":" key ":" value)
key             = label / oid
value           = 1*idchar
label           = "CN" / "L" / "ST" / "O" / "OU" / "C" / "STREET"
oid             = 1*DIGIT *("." 1*DIGIT)
```

Figure 4: ABNF Definition of Subject Policy

<key>:<value> are the subject name fields in chain[0].subject in any order. Key repetitions are not allowed. Values must be percent-encoded.

Example:

did:x509:0:sha256:WE..jk::subject:C:US:ST:Texas:L:Austin:O:Example

Rego policy:

```
validate_predicate(name, value) := true if {
    name == "subject"
    items := split(value, ":")
    count(items) % 2 == 0
    subject := {k: v |
        some i
        i % 2 == 0
        k := items[i]
        v := urlquery.decode(items[i+1])
    }
    count(subject) >= 1
    object.subset(input.chain[0].subject, subject) == true
}
```

Figure 5: Rego Function Validating Subject Policy

2.3. "san" predicate

```
predicate-name  = "san"
predicate-value = san-type ":" san-value
san-type        = "email" / "dns" / "uri"
san-value       = 1*idchar
```

Figure 6: ABNF Definition of SAN Policy

san-type is the SAN type and must be one of email, dns, or uri. Note that dn is not supported.

san-value is the SAN value, percent-encoded.

The pair [<san_type>, <san_value>] is one of the items in chain[0].extensions.san.

Example:

did:x509:0:sha256:WE..jk::san:email:bob%40example.com

Rego policy:

```
validate_predicate(name, value) := true if {
    name == "san"
    [san_type, san_value_encoded] := split(value, ":")
    san_value := urlquery.decode(san_value_encoded)
    [san_type, san_value] == input.chain[0].extensions.san[_]
}
```

Figure 7: Rego Function Validating SAN Policy

2.4. "eku" predicate

```
predicate-name  = "eku"
predicate-value = eku
eku             = oid
oid             = 1*DIGIT *("." 1*DIGIT)
```

Figure 8: ABNF Definition of EKU Policy

eku is one of the OIDs within chain[0].extensions.eku.

Example:

did:x509:0:sha256:WE..jk::eku:1.3.6.1.4.1.311.10.3.13

Rego policy:

```
validate_predicate(name, value) := true if {
    name == "eku"
    value == input.chain[0].extensions.eku[_]
}
```

Figure 9: Rego Function Validating EKU Policy

2.5. "fulcio-issuer" predicate

```
predicate-name   = "fulcio-issuer"
predicate-value  = fulcio-issuer
fulcio-issuer    = 1*idchar
```

Figure 10: ABNF Definition of Fulcio-Issuer Policy

fulcio-issuer is chain[0].extensions.fulcio_issuer, without leading https://, percent-encoded.

Example:

did:x509:0:sha256:WE..jk::fulcio-issuer:accounts.google.com::san:email:bob%40example.com

Example 2:

did:x509:0:sha256:WE..jk::fulcio-issuer:issuer.example.com::san:uri:https%3A%2F%2Fexample.com%2Focto-org%2Focto-automation%2Fworkflows%2Foidc.yml%40refs%2Fheads%2Fmain

Rego policy:

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

validate_predicate(name, value) := true if {
    name == "fulcio-issuer"
    suffix := urlquery.decode(value)
    concat("", ["https://", suffix]) == input.chain[0].extensions.\
                                                          fulcio_issuer
}
```

Figure 11: Rego Function Validating Fulcio-Issuer Policy

2.6. DID resolution options

This DID method introduces a new DID resolution option called x509chain:

Name: x509chain

Value type: string

The value is constructed as follows:

1. Encode each certificate C that is part of the chain as the string b64url(DER(C)).

2. Concatenate the resulting strings in order, separated by comma ",".

3. Example DID Document

This illustrates what a typical DID document ([DID-DOCUMENT]), describing the DID subject and the methods it can use to authenticate itself, can look like once resolved:

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:x509:0:sha256:hH..GE::subject:CN:Example",
  "verificationMethod": [
    {
      "id": "did:x509:0:sha256:hH..GE::subject:CN:Example#key-1",
      "type": "JsonWebKey2020",
      "controller": "did:x509:0:sha256:hH..GE::subject:CN:Example",
      "publicKeyJwk": {
        "kty": "RSA",
        "n": "s9..WQ",
        "e": "AQAB"
      }
    }
  ],
  "assertionMethod": [
    "did:x509:0:sha256:hH..GE::subject:CN:Example#key-1"
  ],
  "keyAgreement": [
    "did:x509:0:sha256:hH..GE::subject:CN:Example#key-1"
  ]
}
```

Figure 12: JSON Controller Document Example

4. CDDL for a JSON Data Model for X.509 Certificate Chains

```
CertificateChain = [2*Certificate]  ; leaf is first

Certificate = {
    fingerprint: {
        ; base64url-encoded hashes of the DER-encoded certificate
        sha256: base64url,     ; FIPS 180-4, SHA-256
        sha384: base64url,     ; FIPS 180-4, SHA-384
        sha512: base64url      ; FIPS 180-4, SHA-512
    },
    issuer: Name,              ; RFC 5280, Section 4.1.2.4
    subject: Name,             ; RFC 5280, Section 4.1.2.6
    extensions: {
        ? eku: [+OID],         ; RFC 5280, Section 4.2.1.12
        ? san: [+SAN],         ; RFC 5280, Section 4.2.1.6
        ? fulcio_issuer: tstr
        ; http://oid-info.com/get/1.3.6.1.4.1.57264.1.1
    }
}

; X.509 Name as an object of attributes
; Repeated attribute types are not supported
```

```
  ; Common attribute types have human-readable labels (see below)
  ; Other attribute types use dotted OIDs
  ; Values are converted to UTF-8
  Name = {
      ; See RFC 4514, Section 3, for meaning of common attribute types
      ? CN: tstr,
      ? L: tstr,
      ? ST: tstr,
      ? O: tstr,
      ? OU: tstr,
      ? C: tstr,
      ? STREET: tstr,
      * OID => tstr
  }

  ; base64url-encoded data, see RFC 4648, Section 5
  base64url = tstr

  ; ASN.1 Object Identifier
  ; Dotted string, for example "1.2.3"
  OID = tstr

  ; X.509 Subject Alternative Name
  ; Strings are converted to UTF-8
  SAN = rfc822Name / DNSName / URI / DirectoryName
  rfc822Name = ["email", tstr] ; Example: ["email", "user@example.com"]
  DNSName = ["dns", tstr]      ; Example: ["dns", "example.com"]
  URI = ["uri", tstr]          ; Example: ["uri", "https://example.com"]
  DirectoryName = ["dn", Name] ; Example: ["dn", {CN: "Example"}]
```

Figure 13: CDDL Definition of did:x509 JSON Data Model

5. Security Consideration

5.1. Identifier Ambiguity

This DID method maps characteristics of X.509 certificate chains to identifiers. It allows a single identifier to map to multiple certificate chains, giving the identifier stability across the expiry of individual chains. However, if the policies used in the identifier are chosen too loosely, the identifier may match too wide a set of certificate chains. This may have security implications as it may authorize an identity for actions it was not meant to be authorized for.

To mitigate this issue, the certificate authority should publish their expected usage of certificate fields and indicate which ones constitute a unique identity, versus any additional fields that may

be of an informational nature. This will help users create an appropriate did:x509 as well as consumers of signed content to decide whether it is appropriate to trust a given did:x509.

5.2. X.509 Trust Stores

Typically, a verifier trusts an X.509 certificate by applying chain validation defined in Section 6 of [RFC5280] using a set of certificate authority (CA) certificates as trust store, together with additional application-specific policies.

This DID method does not require an X.509 trust anchor store but rather relies on verifiers either trusting an individual DID directly or using third-party endorsements for a given DID, like [VC], to establish trust.

By layering this DID method on top of X.509, verifiers are free to use traditional chain validation (for example, verifiers unaware of DID), or rely on DID as an ecosystem to establish trust.

5.3. Use of Identifier Contents

While it is acceptable to use a did:x509 identifier as an opaque handle when it has been endorsed through an external trust mechanism, such as a verifiable credential or a trusted registry, implementers MUST NOT parse or interpret individual components of the identifier string for authorization decisions unless the identifier has been resolved against a verified certificate chain.

Specifically, extracting and relying upon subject names, organizational information, or other embedded values directly from the identifier string, without performing full resolution and chain validation, is insecure. An attacker could craft a syntactically valid did:x509 identifier containing arbitrary values that do not correspond to any legitimate certificate chain. Only after successful resolution, which includes verification of the CA fingerprint against the provided chain and validation of all policy predicates, can the identifier be considered authentic. Systems that bypass this resolution process and instead parse identifier components directly are vulnerable to impersonation and privilege escalation attacks.

6. IANA Considerations

// RFC Editor: Please replace "RFCthis" with the RFC number assigned to this document.

// RFC Editor: Some considerations

7. References

7.1. Normative References

[BCP26] 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>.

[DID] "W3C DID v1.0 specification", n.d., <https://www.w3.org/TR/did-1.0/>.

[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>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

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

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

[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>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/rfc/rfc8610>.

[RFC9165] Bormann, C., "Additional Control Operators for the Concise Data Definition Language (CDDL)", RFC 9165, DOI 10.17487/RFC9165, December 2021, <https://www.rfc-editor.org/rfc/rfc9165>.

[STD90] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[STD94] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[VC] "W3C Verifiable Credentials", n.d., <https://www.w3.org/TR/vc-data-model/>.

7.2. Informative References

[DID-DOCUMENT] "DID Document Definition", n.d., <https://www.w3.org/TR/did-1.0/#dfn-did-documents>.

[I-D.ietf-scitt-architecture] Birkholz, H., Delignat-Lavaud, A., Fournet, C., Deshpande, Y., and S. Lasker, "An Architecture for Trustworthy and Transparent Digital Supply Chains", Work in Progress, Internet-Draft, draft-ietf-scitt-architecture-22, 10 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-scitt-architecture-22>.

[REGO] "Rego", n.d., <https://www.openpolicyagent.org/docs/latest/policy-language/>.

[RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <https://www.rfc-editor.org/rfc/rfc8792>.

[RFC9360] Schaad, J., "CBOR Object Signing and Encryption (COSE): Header Parameters for Carrying and Referencing X.509 Certificates", RFC 9360, DOI 10.17487/RFC9360, February 2023, <https://www.rfc-editor.org/rfc/rfc9360>.

[RFC9597] Looker, T. and M.B. Jones, "CBOR Web Token (CWT) Claims in COSE Headers", RFC 9597, DOI 10.17487/RFC9597, June 2024, <https://www.rfc-editor.org/rfc/rfc9597>.

Acknowledgments

The authors would like to thank _list_ for their reviews and suggestions.

Authors' Addresses

Maik Riechert
Microsoft
Email: Maik.Riechert@microsoft.com

Antoine Delignat-Lavaud
Microsoft
Email: antdl@microsoft.com

Henk Birkholz
Fraunhofer SIT
Email: henk.birkholz@ietf.contact

Amaury Chamayou
Microsoft
Email: Amaury.Chamayou@microsoft.com