Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 23 April 2026

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

20 October 2025

x509 Decentralized Identifier
draft-birkholz-did-x509-00

Abstract

Some abstract

Status of This Memo

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

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

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

This Internet-Draft will expire on 23 April 2026.

Table of Contents

1. Introduction

This document aims to define an interoperable and flexible issuer identifier 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]. This issuer identifier is convenient for references and policy evaluation, for example in the context of transparency ledgers.

2. 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].

3. Identifier Syntax

The did:x509 ABNF definition defined below uses the syntax defined in [RFC5234] and the corresponding definitions for ALPHA and DIGIT. The [DIDV1] contains the definition for idchar.

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

In this draft, version is 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. policy-name is a policy name and policy-value is a policy-specific value. :: is used to separate multiple policies from each other.

The following sections define the policies and their policy-specific syntax.

Validation of policies is formally defined using [REGO] policies, though there is no expectation that implementations use Rego.

The input to the Rego engine is the JSON document {"did": "<DID>", "chain": <CertificateChain>}.

Core Rego policy:

```
parse_did(did) := [ca_fingerprint_alg, ca_fingerprint, policies] 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], ":")
    policies_raw := array.slice(parts, 1, count(parts))
    policies := [y |
        some i
        s := policies_raw[i]
        j := indexof(s, ":")
        y := [substring(s, 0, j), substring(s, j+1, -1)]
    ]
}

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

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_policy function.

3.1. Percent-encoding

Some of the policies 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 / "-" / "." / "_"
```

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

3.2. "subject" policy

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

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

Example:

```
did:x509:0:sha256:WE4P5dd8DnLHSkyHaIjhp4udlkF9LqoKwCvu9gl38jk::subjec
t:C:US:ST:California:L:San%20Francisco:O:GitHub%2C%20Inc.
```

Rego policy:

```
validate_policy(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
}
```

3.3. "san" policy

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

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:WE4P5dd8DnLHSkyHaIjhp4udlkF9LqoKwCvu9gl38jk::san:em
ail:bob%40example.com
```

Rego policy:

```
validate_policy(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[_]
}
```

3.4. "eku" policy

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

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

Example:

```
did:x509:0:sha256:WE4P5dd8DnLHSkyHaIjhp4udlkF9LqoKwCvu9gl38jk::eku:1.
3.6.1.4.1.311.10.3.13
```

Rego policy:

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

3.5. "fulcio-issuer" policy

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

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

Example:

```
did:x509:0:sha256:WE4P5dd8DnLHSkyHaIjhp4udlkF9LqoKwCvu9gl38jk::fulcio
-issuer:accounts.google.com::san:email:bob%40example.com
```

Example 2:

```
did:x509:0:sha256:WE4P5dd8DnLHSkyHaIjhp4udlkF9LqoKwCvu9gl38jk::fulcio
-issuer:token.actions.githubusercontent.com::san:uri:https%3A%2F%2Fgi
thub.com%2Focto-org%2Focto-
automation%2F.github%2Fworkflows%2Foidc.yml%40refs%2Fheads%2Fmain
```

Rego policy:

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

3.6. 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 ",".

4. Example Controller Document

The illustrates what a typical Controller document can look like once resolved:

```
{
  "@context": "https://www.w3.org/ns/did/v1",
  "id": "did:x509:0:sha256:hH32p4SXlD8n_HLrk_mmNzIKArVh0KkbCeh6eAftfGE::subject:CN:Micros
oft%20Corporation",
  "verificationMethod": [
    {
      "id": "did:x509:0:sha256:hH32p4SXlD8n_HLrk_mmNzIKArVh0KkbCeh6eAftfGE::subject:CN:Mi
crosoft%20Corporation#key-1",
      "type": "JsonWebKey2020",
      "controller": "did:x509:0:sha256:hH32p4SXlD8n_HLrk_mmNzIKArVh0KkbCeh6eAftfGE::subje
ct:CN:Microsoft%20Corporation",
      "publicKeyJwk": {
        "kty": "RSA",
        "n": "s9HduD2rvmO-SGksB4HR-qvSK379St8NnUZBH8xBiQvt2zONOLUHWQibeBW4NLUfHfzMaOM77Rh
NlqPNiDRKhChlG1aHqEHSAaQBGrmr0ULGIzq-1YvqQufMGYBFfq0sc10UdvWqT0RjwkPQTu4bjg37zSYF9OcGxS9u
GnPMdWRM0ThOsYUcDmMoCaJRebsLUBpMmYXkcUYXJrcSGAaUNd0wjhwIpEogOD-AbWW_7TPZOl-JciMj40a78EEXI
c2p06lWHfe5hegQ7uGIlSAPG6zDzjhjNkzE63_-GoqJU-6QLazbL5_y27ZDUAEYJokbb305A-dOp930CjTar3BvWQ
",
        "e": "AQAB"
      }
    }
  ],
  "assertionMethod": [
    "did:x509:0:sha256:hH32p4SXlD8n_HLrk_mmNzIKArVh0KkbCeh6eAftfGE::subject:CN:Microsoft%
20Corporation#key-1"
  ],
  "keyAgreement": [
    "did:x509:0:sha256:hH32p4SXlD8n_HLrk_mmNzIKArVh0KkbCeh6eAftfGE::subject:CN:Microsoft%
20Corporation#key-1"
  ]
}
```


Figure 1: JSON controller document example

5. CDDL for a JSON Data Model for X.509 Certification Paths

```
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", "bill@microsoft.com"]
DNSName = ["dns", tstr]      ; Example: ["dns", "microsoft.com"]
URI = ["uri", tstr]          ; Example: ["uri", "https://microsoft.com"]
DirectoryName = ["dn", Name] ; Example: ["dn", {CN: "Microsoft"}]
```

Figure 2: CDDL definition of did:x.509 JSON Data Model

6. Privacy Considerations

Some considerations

7. Security Consideration

7.1. 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.

7.1.1. 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.

8. IANA Considerations

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

// RFC Editor: Some considerations

9. References

9.1. 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>.

[DIDV1] "W3C DID v1.0 specification", n.d., <https://www.w3.org/TR/2022/REC-did-core-20220719/>.

[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/>.

9.2. Informative References

[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/>.

[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