

Network Working Group
Internet-Draft
Intended status: Informational
Expires: 1 September 2026

S. Nandakumar
C. Jennings
Cisco
28 February 2026

SD Agent: Selective Disclosure for Agent Discovery and Identity Management
draft-nandakumar-agent-sd-jwt-02

Abstract

This document defines how Selective Disclosure for JWTs (SD-JWT) integrates with Agent-to-Agent (A2A) systems to provide privacy-preserving agent discovery and cryptographically verifiable identity management. It specifies the SD-Card format - an SD-JWT encoding of Agent Cards that enables selective disclosure of agent capabilities, contact information, and operational metadata while maintaining cryptographic integrity and preventing correlation across different interaction contexts.

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 1 September 2026.

Copyright Notice

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

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

Table of Contents

1. Introduction
 - 1.1. Terminology
 - 1.2. Notational Conventions
2. Overview of SD-Card Architecture and Benefits
 - 2.1. System Architecture
 - 2.2. Challenges with Traditional Agent Discovery
3. SD-Card Agent Card
 - 3.1. Agent Card Structure
 - 3.2. Selective Disclosure Claims

- 3.2.1. Skills and Capabilities
 - 3.2.2. Additional Agent Interfaces
 - 3.2.3. Agent Capabilities
 - 3.2.4. Security Schemes and Authentication
 - 3.2.5. Provider Information
 - 3.2.6. Input and Output Modes
 - 3.2.7. Documentation and Support
 - 3.3. Agent Card Issuance
 - 3.3.1. Issuance Process Overview
 - 4. Agent Discovery Protocols with Privacy Protection
 - 4.1. Well-Known URI Discovery
 - 4.2. Registry-Based Discovery
 - 4.3. Contextual Discovery
 - 4.4. Integration with Model Context Protocol
 - 4.4.1. Protocol Layering
 - 5. Cryptographic Authentication and Capability-Based Authorization
 - 5.1. SD-JWT with Key Binding
 - 5.2. Agent Authentication Process
 - 5.2.1. Step 1: Agent Card Presentation
 - 5.2.2. Step 2: Key Binding JWT Creation
 - 5.2.3. Step 3: Cryptographic Verification
 - 5.3. Advanced Authorization Flows
 - 5.3.1. Capability-Based Discovery
 - 5.3.2. Role-Based Disclosure
 - 5.3.3. Token Exchange for Sub-Agent Delegation
 - 6. IANA Considerations
 - 6.1. Media Type Registration
 - 6.2. Verifiable Credential Type Registration
 - 6.3. JSON Web Token Claims Registration
 - 7. Security Considerations
 - 7.1. Token Binding and Replay Prevention
 - 7.2. Agent Identity Verification
 - 7.3. Delegation and Multi-Agent Scenarios
 - 7.4. Privacy Considerations
 - 7.4.1. Correlation Prevention
 - 7.4.2. Minimal Disclosure Principle
 - 7.4.3. Data Minimization in Agent Cards
 - 7.5. Transport Security
 - 7.6. Issuer Security
 - 7.7. Cryptographic Agility
 - 8. References
 - 8.1. Normative References
 - 8.2. Informative References
- Authors' Addresses

1. Introduction

The Agent-to-Agent (A2A) specification [A2A-SPEC] defines protocols for autonomous agent discovery and interaction through standardized Agent Cards. Current A2A implementations have following limitations:

1. Information Leakage: Agent Cards expose all capabilities during discovery
2. Weak Authentication: No cryptographic verification of agent identity
3. Static Disclosure: No context-based capability filtering
4. Linkability: Identical presentations enable cross-context tracking

This document specifies SD-Card, an SD-JWT [RFC9901] encoding of Agent Cards that addresses these limitations through:

- * Selective Disclosure: Context-specific capability revelation

- * Cryptographic Authentication: Key-bound agent identity verification
- * Unlinkable Presentations: Privacy-preserving multi-context interactions
- * Backward Compatibility: Interoperability with existing A2A implementations

1.1. 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.

Agent Card: A JSON metadata document describing an agent's capabilities, identity, and interaction requirements as defined in [A2A-SPEC].

SD-Card: An Agent Card encoded as an SD-JWT that enables selective disclosure of agent metadata.

Agent Registry: A trusted service that issues and manages SD-Cards for agents within a domain or federation.

Discovery Context: A named security context (e.g., "public", "internal", "diagnostic") that determines which claims are disclosed in an SD-Card presentation.

Capability Disclosure: The selective revelation of specific agent capabilities based on the requesting entity's authorization and context.

Key Binding: Cryptographic proof that the presenter of an SD-Card possesses the private key corresponding to the public key in the SD-Card's cnf claim.

SD-JWT Media Type: The application/vnd.sd-jwt+json media type is used to indicate content that contains Selective Disclosure JWT data within JSON structures. This vendor-specific media type distinguishes SD-JWT presentations from standard JSON content and signals that the receiver should process the content using SD-JWT parsing and verification procedures.

1.2. Notational Conventions

This document follows SD-JWT [RFC9901] conventions:

- * Base64url encoding for binary data
- * JWS format for signed tokens
- * SHA-256 hashing for selective disclosure
- * Tilde (~) separator for SD-JWT serialization:

<Issuer-signed JWT>~<Disclosure 1>~<Disclosure 2>~...<Key Binding JWT>

2. Overview of SD-Card Architecture and Benefits

SD-Card enhances Agent-to-Agent (A2A) interactions by providing privacy-preserving agent discovery and cryptographically verifiable agent authentication. SD-Card enables selective disclosure of agent

capabilities, skills, and endpoints based on requester authorization and discovery context, unlinkable agent interactions across different contexts, cryptographic integrity of agent metadata, and key binding for authentication that complements transport-layer security.

2.1. System Architecture

The SD-Card architecture involves four primary components: Agent Registries that issue credentials, Agents that hold and present SD-Cards, Services that agents interact with, and Clients that discover and invoke agents. The following diagram illustrates these relationships and the flow of SD-Card issuance and presentation:

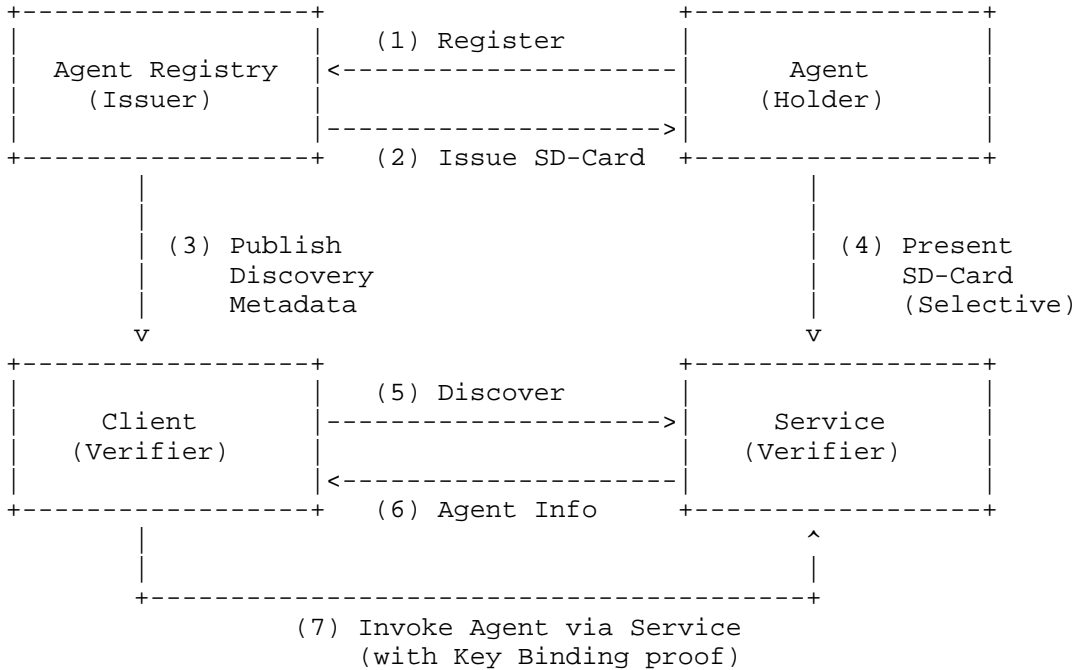


Figure 1: SD-Card System Architecture

The architecture operates as follows:

1. **Registration:** An Agent registers with an Agent Registry, providing its capabilities, skills, and public key material for key binding.
2. **SD-Card Issuance:** The Registry issues an SD-Card (SD-JWT) containing the agent's metadata with selectively disclosable claims. The SD-Card includes the agent's confirmation key (cnf claim) for holder binding.
3. **Discovery Publication:** The Registry may publish non-sensitive discovery metadata to enable clients to locate agents matching their requirements.
4. **Selective Presentation:** When interacting with Services, the Agent presents its SD-Card with only the disclosures relevant to the interaction context. Different contexts receive different subsets of claims.
5. **Client Discovery:** Clients query the Registry or Services to discover agents with specific capabilities. The discovery response contains only claims appropriate for the discovery context.
6. **Agent Information:** The discovered agent information is returned to the client, enabling informed decisions about which agent to

invoke.

7. Agent Invocation: The Client invokes the Agent through the Service, with the Agent providing a Key Binding JWT to prove possession of the private key corresponding to its SD-Card.

This architecture enables privacy-preserving agent discovery while maintaining cryptographic verifiability. Agents control what information they reveal in each context, and verifiers can trust that presented claims are authentic and unmodified.

2.2. Challenges with Traditional Agent Discovery

Traditional agent discovery in Agent-to-Agent protocol faces several challenges:

Information Leakage: Agent Cards expose all capabilities to any entity performing discovery, potentially revealing sensitive operational details.

Linkability: Identical capabilities presented to different verifiers enable tracking across contexts.

Authenticity: Without cryptographic verification, Agent Cards can be forged or tampered with during transmission.

Capability Enumeration: Attackers can enumerate all agent capabilities by performing discovery, potentially identifying attack vectors.

Context Insensitivity: Agents cannot adapt capability disclosure based on the requesting entity's authorization level or intended use case.

Allowing agents to selectively disclose their capabilities addresses these challenges through:

Selective Capability Disclosure: Agents reveal only capabilities relevant to the specific discovery context or requester authorization level.

Unlinkable Presentations: Different presentations of the same agent to different verifiers cannot be linked, enhancing privacy.

Flexible Authentication: Key binding enables strong agent authentication without relying solely on transport security.

Batch Credential Issuance: Multiple context-specific credentials can be issued to the same agent for different use cases.

3. SD-Card Agent Card

This specification defines "SD-Card". An SD-Card is an Agent Card encoded as an SD-JWT.

3.1. Agent Card Structure

The base Agent Card structure follows the A2A specification [A2A-SPEC] but is extended to support selective disclosure. The card consists of three main components:

1. Base Claims: Always disclosed information that establishes the agent's identity, protocol version, and basic metadata
2. Selective Disclosure Claims: A2A-specific information that can be selectively revealed based on the discovery context including

skills, interfaces, and provider details

3. Cryptographic Binding: Keys and signatures that ensure authenticity and enable secure agent interactions

The Agent Card is represented as a Selective Disclosure JSON Web Token (SD-JWT) that contains both standard A2A Agent Card fields and selective disclosure metadata :

```
{
  "iss": "https://registry.example.com",
  "sub": "agent:georoute-planner-v1",
  "iat": 1704063600,
  "exp": 1735685999,
  "vct": "urn:ietf:params:oauth:token-type:sd-agent-card",
  "_sd_alg": "sha-256",
  "_sd": [
    "kV7i-VgPK9Qj8vYNJ4L8hFG3cR9ZqX2mE5wC6oAlnBs", // skills
    "3sdf8mN2p9uX7L3vE8nGrR5kW6oF4tC9jL7vM2nX8qE", // additionalInterfaces
    "8kF2nM3p4uY9L5cG6rT7qW9oR2vX1jE4mN8sC5fH0aZ", // capabilities
    "9mH4pL5rZ8wQ2nX7cV6sT3kE1oY0jF9uR4vB6nM8fG2", // securitySchemes
    "7kRt2qW5mX9jN4cF7zPsE8nG1vH0oY3lM6xC9fK2bA8", // provider
    "5nK3wL8tB4eR9mY2cF6qG7oT1vX0jH3pN4rS8zE5fC6", // defaultInputModes
    "2pM8fT4nR6oX3cG7zE9qL5wY1vS0jF6kH2nB8mC4rT9" // defaultOutputModes
  ],
  "protocolVersion": "0.2.9",
  "name": "GeoSpatial Route Planner Agent",
  "description": "Provides advanced route planning and traffic analysis services",
  "version": "1.2.0",
  "url": "https://georoute-agent.example.com/a2a/v1",
  "cnf": {
    "jwk": {
      "kty": "EC",
      "crv": "P-256",
      "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
      "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
    }
  }
}
```

The following claims are permanently disclosed in the SD-Card:

iss: The issuer of the Agent Card (typically an Agent Registry). This establishes the trust anchor for the credential and enables signature verification. The issuer **MUST** be a resolvable URI that provides access to the issuer's public key material.

sub: The unique identifier of the agent within the issuer's namespace. This identifier **MUST** be stable across different Agent Card issuances for the same logical agent, enabling correlation when authorized while supporting unlinkable presentations through different salt values.

iat: The time at which the Agent Card was issued (Unix timestamp).

exp: The expiration time of the Agent Card (Unix timestamp).

vct: The verifiable credential type, identifying this as an SD-Card Agent Card. This value **MUST** be "urn:ietf:params:oauth:token-type:sd-agent-card" for compliance with this specification.

name: The human-readable name of the agent.

description: A brief description of the agent's purpose and primary function. This helps users understand the agent's role without revealing sensitive capability details.

version: The version of the agent software.

cnf: The confirmation key for agent authentication. This public key enables key binding and ensures that only the legitimate agent can present the Agent Card. The key MUST be in JSON Web Key (JWK) format.

3.2. Selective Disclosure Claims

The following claims are selectively disclosed based on the discovery context:

3.2.1. Skills and Capabilities

The skills claim contains an array of A2A AgentSkill objects that the agent can perform. Each skill definition follows the A2A specification and includes input/output modes, examples, and metadata for proper integration.

3.2.1.1. Original Skills JSON Object:

```
"skills": [
  {
    "id": "route-optimizer-traffic",
    "name": "Traffic-Aware Route Optimizer",
    "description": "Calculates optimal driving route between locations with real-time traffic",
    "tags": ["maps", "routing", "navigation", "directions", "traffic"],
    "examples": [
      "Plan a route from '1600 Amphitheatre Parkway, Mountain View, CA' to 'San Francisco International Airport' avoiding tolls.",
      "{\\"origin\\": {\\"lat\\": 37.422, \\"lng\\": -122.084}, \\"destination\\": {\\"lat\\": 37.7749, \\"lng\\": -122.4194}, \\"preferences\\": [\\"avoid_ferries\\"]}"
    ],
    "inputModes": ["application/json", "text/plain"],
    "outputModes": [
      "application/json",
      "application/vnd.geo+json",
      "text/html"
    ]
  },
  {
    "id": "custom-map-generator",
    "name": "Personalized Map Generator",
    "description": "Creates custom map images based on user-defined points of interest",
    "tags": ["maps", "customization", "visualization", "cartography"],
    "examples": [
      "Generate a map of my upcoming road trip with all planned stops highlighted.",
      "Show me a map visualizing all coffee shops within a 1-mile radius of my current location."
    ],
    "inputModes": ["application/json"],
    "outputModes": [
      "image/png",
      "image/jpeg",
      "application/json",
      "text/html"
    ]
  }
]
```

3.2.1.2. SD-JWT Transformation Process:

When converting the skills claim to selective disclosure, the Agent Registry performs the following transformation:

Step 1: Generate Salt Value

A cryptographically secure random salt is generated for the skills claim:

Salt: "_26bc4LT-ac6q2KI6cBW5es"

Step 2: Create Disclosure Array

The skills claim is packaged into a disclosure array format:

```
[
  "_26bc4LT-ac6q2KI6cBW5es",
  "skills",
  [
    {
      "id": "route-optimizer-traffic",
      "name": "Traffic-Aware Route Optimizer",
      "description": "Calculates optimal driving route between locations with real
-time traffic",
      "tags": ["maps", "routing", "navigation", "directions", "traffic"],
      "examples": [
        "Plan a route from '1600 Amphitheatre Parkway, Mountain View, CA' to 'San
Francisco International Airport' avoiding tolls.",
        "{ \"origin\": { \"lat\": 37.422, \"lng\": -122.084 }, \"destination\": { \"la
t\": 37.7749, \"lng\": -122.4194 }, \"preferences\": [ \"avoid_ferries\" ] }"
      ],
      "inputModes": ["application/json", "text/plain"],
      "outputModes": [
        "application/json",
        "application/vnd.geo+json",
        "text/html"
      ]
    },
    {
      "id": "custom-map-generator",
      "name": "Personalized Map Generator",
      "description": "Creates custom map images based to user-defined points of in
terest",
      "tags": ["maps", "customization", "visualization", "cartography"],
      "examples": [
        "Generate a map of my upcoming road trip with all planned stops highlighte
d.",
        "Show me a map visualizing all coffee shops within a 1-mile radius of my c
urrent location."
      ],
      "inputModes": ["application/json"],
      "outputModes": [
        "image/png",
        "image/jpeg",
        "application/json",
        "text/html"
      ]
    }
  ]
]
```

Step 3: Base64url Encode Disclosure

The disclosure array is JSON-serialized and base64url-encoded:

```
WyJfMjZiYzRMVClhYzZxMktJNmNCVzVlcyIsInNraWxscyIsW3siaWQiOiJyb3V0ZS1vcHRp
bWl6ZXItbWZpYyIsIm5hbWUiOiJUCmFmZmljLUF3YXJlIFJvdXRlIE9wdGltaxplciIs
ImRlc2NyaXB0aW9uIjoie2F5Y3VsYXRlcjBvcHRpbWFSIGRyaXZpbmcgc29ldGUgYmV0d2Vl
biBsb2NhdGlvbnMgd2l0aCBzZWFSLXRpbWUgdHJhZmZpYyIsInRhZ3MiOlsibWFwcyIsInJv
```


dXRpbmciLCJuYXZpZ2F0aW9uIiwizGlyZWN0aW9ucyIsInRyYWZmaWMiXSwizXhhbXBsZXMi
OlsiUGxhb20gJzE2MDAgQWlwaGl0aGVhdHJlIFBhcmt3YXksIElvdW50
YWluIFZpZzXcsIENBjYB0byAnU2FuIEZyYW5jaXNjb25hdGlvbmFsIEFpcnBvcnQn
IGF2b2lkaW5nIHRvbGxzLiIsIntcIm9yaWdpblwiOiB7XCJSYXRcIjogMzcuNDIyLCBcImxu
ZlwiOiAtMTIyLjA4NH0sIFwiZGVzdGluYXRpb25cIjogelwibGF0XCI6IDM3Ljc3NDksIFwi
bG5nXCI6IC0xMjIuNDU5NH0sIFwicHJlZmVzZW5jZXRcIjogWlwiYXZvaWRfZmVycmllclwi
XX0iXSwiaW5wdXRnb2RlcYI6WyJhcHBsaWNhdGlvbi9qc29uIiwidGV4dC9wbGFpbiJdLCJv
dXRwdXRnb2RlcYI6WyJhcHBsaWNhdGlvbi9qc29uIiwidGV4dC9wbGFpbiJdLCJv
c29uIiwidGV4dC9odGlsIl19LHsiaWQiOiJjdXN0b20tbWFWLWdlbmVyYXRvciiIsIm5hbWUi
OiJQZzJzb25hbG16ZWQgTWFWIEdlbmVyYXRvciiIsImRlc2NyaXB0aW9uIjoIQ3JlYXRlcYBj
dXN0b20gbWFWIGltYWdlcyBiYXNlZCBvbiBlc2VyLWRLZmluZWQgcG9pbnRzIG9mIGludGVy
ZXN0IiwidGFncyI6WyJtYXBzIiwidGV4dG9taXphdGlvbiIsInZpc3VhbG16YXRpb24iLCJj
YXJ0b2dyYXB0eSJdLCJleGFtcGxlcyI6WyJHZW5lcmF0ZSBhIGlhcCBvZiBteSBlcGNvbWlu
ZyByb2FkIHRYaXAgd2l0aCBhbGwgcGxhbm5lZCBzdG9wcyBoaWdobGlnaHRlZC4iLCJTaG93
IG1lIGEGbWFWIHZpc3VhbG16aW5nIGFsbCBjb2ZmZWUgc2hvcHMgd2l0aGluIGEGMS1taWxl
IHJhZG1lcYBvZiBteSBjdXJyZW50IGxvY2F0aW9uLiJdLCJpbnBldElvZGVzIjpbImFwcGxp
Y2F0aW9uL2pzb24iXSwib3V0cHV0TW9kZXMiOlsiaW1hZ2UvcG5nIiwiaW1hZ2UvanBlZyIs
ImFwcGxpY2F0aW9uL2pzb24iLCJ0ZXh0L2h0bWwiXX1dXQ

Step 4: Calculate SHA-256 Digest

A SHA-256 hash is computed over the base64url-encoded disclosure:

```
digest = SHA256(base64url_encode(disclosure_array))  
        = "kV7i-VgPK9Qj8vYNJ4L8hFG3cR9ZqX2mE5wC6oAlnBs"
```

Step 5: Include Digest in SD-JWT

The digest is included in the `_sd` array of the Issuer-signed JWT payload:

```
{  
  "iss": "https://registry.example.com",  
  "sub": "agent:georoute-planner-v1",  
  "_sd_alg": "sha-256",  
  "_sd": [  
    "kV7i-VgPK9Qj8vYNJ4L8hFG3cR9ZqX2mE5wC6oAlnBs",  
    "7iF2sM8kT5nL9pX3cV6zR4qW1eY8oU0jH3vB7nM2kF9"  
  ],  
  "name": "GeoSpatial Route Planner Agent",  
  "cnf": { }  
}
```

Step 6: Selective Disclosure

When presenting the agent card, the skills disclosure is included only if authorized:

<Issuer-signed JWT>~<Skills Disclosure>~<Other Disclosures>~<Key Binding JWT>

During verification, the verifier:

1. Decodes the skills disclosure from base64url
2. Calculates SHA-256 hash of the decoded disclosure
3. Confirms the hash matches the digest in the `_sd` array
4. Extracts the skills claim value for use

3.2.2. Additional Agent Interfaces

The `additionalInterfaces` claim defines alternative transport mechanisms and endpoints for A2A communication as specified in the `AgentInterface` object structure.

3.2.2.1. Original values

The original additionalInterfaces claim in the agent metadata contains:

```
{
  "additionalInterfaces": [
    {
      "url": "https://georoute-agent.example.com/a2a/v1",
      "transport": "JSONRPC"
    },
    {
      "url": "https://georoute-agent.example.com/a2a/grpc",
      "transport": "GRPC"
    },
    {
      "url": "https://georoute-agent.example.com/a2a/json",
      "transport": "HTTP+JSON"
    }
  ]
}
```

3.2.2.2. SD-JWT Transformation Process:

When converting the additionalInterfaces claim to selective disclosure, the Agent Registry performs the following transformation:

Step 1: Generate Salt Value A cryptographically secure random salt is generated for the additionalInterfaces claim:

Salt: "_26bc4LT-ac6q2KI6cBW5es"

Step 2: Create Disclosure Array The additionalInterfaces claim is packaged into a disclosure array format:

```
[
  "_26bc4LT-ac6q2KI6cBW5es",
  "additionalInterfaces",
  [
    {
      "url": "https://georoute-agent.example.com/a2a/v1",
      "transport": "JSONRPC"
    },
    {
      "url": "https://georoute-agent.example.com/a2a/grpc",
      "transport": "GRPC"
    },
    {
      "url": "https://georoute-agent.example.com/a2a/json",
      "transport": "HTTP+JSON"
    }
  ]
]
```

Step 3: Base64url Encode Disclosure

The disclosure array is JSON-serialized and base64url-encoded:

WyJfmjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsImFkZG10aW9uYWxJbnRlcmZhY2VzIixbeyJlcmwiOiJodHRwczovL2dlb3JvdXRlLWFnZW50LmV4YWlwbGUuY29tL2EyYS92MSIsInRyYW5zcG9ydCI6IkpTT05SUEMifSx7InVybCI6Imh0dHBzOi8vZ2VvcmlldG9tYWdlbnQuZXhhbXBsZS5jb20vYTJhL2dycGMiLCJ0cmFuc3BvcnQiOiJHUlBDIn0seyJlcmwiOiJodHRwczovL2dlb3JvdXRlLWFnZW50LmV4YWlwbGUuY29tL2EyYS9qc29uIiwidHJhbnNwb3J0IjoisFRUUCtKU090InldXQ

Step 4: Calculate SHA-256 Digest

A SHA-256 hash is computed over the base64url-encoded disclosure:

```
digest = SHA256(base64url_encode(disclosure_array))
        = "3sdf8mN2p9uX7L3vE8nGrR5kW6oF4tC9jL7vM2nX8qE"
```

Step 5: Include Digest in SD-JWT

The digest is included in the `_sd` array of the Issuer-signed JWT payload:

```
{
  "iss": "https://registry.example.com",
  "sub": "agent:georoute-planner-v1",
  "_sd_alg": "sha-256",
  "_sd": [
    "3sdf8mN2p9uX7L3vE8nGrR5kW6oF4tC9jL7vM2nX8qE"
  ]
}
```

3.2.3. Agent Capabilities

The capabilities claim provides information about the agent's A2A protocol feature support as defined in the `AgentCapabilities` object.

3.2.3.1. Original values

The original capabilities claim in the agent metadata contains:

```
{
  "capabilities": {
    "streaming": true,
    "pushNotifications": true,
    "stateTransitionHistory": false,
    "extensions": [
      {
        "name": "advanced-routing",
        "version": "1.0",
        "description": "Enhanced route optimization with machine learning"
      }
    ]
  }
}
```

3.2.3.2. SD-JWT Transformation Process:

When converting the capabilities claim to selective disclosure, the Agent Registry performs the following transformation:

Step 1: Generate Salt Value

A cryptographically secure random salt is generated for the capabilities claim:

Salt: `"_26bc4LT-ac6q2KI6cBW5es"`

Step 2: Create Disclosure Array

The capabilities claim is packaged into a disclosure array format:

```
[
  "_26bc4LT-ac6q2KI6cBW5es",
  "capabilities",
  {
    "streaming": true,
    "pushNotifications": true,
    "stateTransitionHistory": false,

```

```

    "extensions": [
      {
        "name": "advanced-routing",
        "version": "1.0",
        "description": "Enhanced route optimization with machine learning"
      }
    ]
  }
}

```

Step 3: Base64url Encode Disclosure

The disclosure array is JSON-serialized and base64url-encoded:

```

WyJfMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsImNhcGFiaWxp dGllcyIseyJzdHJlYWlpbmciOnRydWUsInBlc2hOb
3RpZmljYXRpb25zIjpo0cnVlLCJzdGF0ZVRyYW5zaXRpb25IaXN0b3J5IjpmYWxzZSwiZXh0ZW5zaW9ucyI6W3sibm
FtZSI6ImFkd mFuY2VkLXJvdXRpbmciLCJ2ZXJzaW9uIjo iMS4wIiwizGVzY3JpcHRpb24iOiJFbmhhbmNlZCB yb3V
0ZSBvcHRpbWl6YXRpb24gd2l0aCBtYWNoaW5lIGx1YXJuaW5nInl dfV0

```

Step 4: Calculate SHA-256 Digest

A SHA-256 hash is computed over the base64url-encoded disclosure:

```

digest = SHA256(base64url_encode(disclosure_array))
        = "8kF2nM3p4uY9L5cG6rT7qW9oR2vX1jE4mN8sC5fH0aZ"

```

Step 5: Include Digest in SD-JWT

The digest is included in the `_sd` array of the Issuer-signed JWT payload:

```

{
  "iss": "https://registry.example.com",
  "sub": "agent:georoute-planner-v1",
  "_sd_alg": "sha-256",
  "_sd": [
    "8kF2nM3p4uY9L5cG6rT7qW9oR2vX1jE4mN8sC5fH0aZ"
  ]
}

```

3.2.4. Security Schemes and Authentication

The `securitySchemes` claim defines authentication requirements following A2A SecurityScheme format.

3.2.4.1. Original values

The original `securitySchemes` claim in the agent metadata contains:

```

{
  "securitySchemes": {
    "google": {
      "type": "openIdConnect",
      "openIdConnectUrl": "https://accounts.google.com/.well-known/openid-configuratio
n"
    },
    "apiKey": {
      "type": "apiKey",
      "name": "X-API-Key",
      "in": "header"
    }
  }
}

```

3.2.4.2. SD-JWT Transformation Process:

When converting the securitySchemes claim to selective disclosure, the Agent Registry performs the following transformation:

Step 1: Generate Salt Value

A cryptographically secure random salt is generated for the securitySchemes claim:

Salt: "_26bc4LT-ac6q2KI6cBW5es"

Step 2: Create Disclosure Array

The securitySchemes claim is packaged into a disclosure array format:

```
[
  "_26bc4LT-ac6q2KI6cBW5es",
  "securitySchemes",
  {
    "google": {
      "type": "openIdConnect",
      "openIdConnectUrl": "https://accounts.google.com/.well-known/openid-configuratio
n"
    },
    "apiKey": {
      "type": "apiKey",
      "name": "X-API-Key",
      "in": "header"
    }
  }
]
```

Step 3: Base64url Encode Disclosure

The disclosure array is JSON-serialized and base64url-encoded:

```
WyJmMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsInNlY3VyaXR5U2NoZWllcyIseyJnb29nbGUiOnsidHlwZSI6Im9wZ
W5JZENvbmlY3QiLCJvcGVuSWRDb25uZWNOVXJsIjoiaHR0cHM6Ly9hY2NvdW50cy5nb29nbGUuY29tLy53ZWxsLW
tub3duL29wZW5pZC1jb25maWdlcmF0aW9uIn0sImFwaUtleSI6eyJ0eXB1IjoieXBpS2V5IiwibmFtZSI6IlgtQVB
JLUtleSIsImluIjoiaGVhZGVyInl9XQ
```

Step 4: Calculate SHA-256 Digest

A SHA-256 hash is computed over the base64url-encoded disclosure:

```
digest = SHA256(base64url_encode(disclosure_array))
       = "9mH4pL5rZ8wQ2nX7cV6sT3kE1oY0jF9uR4vB6nM8fG2"
```

Step 5: Include Digest in SD-JWT

The digest is included in the _sd array of the Issuer-signed JWT payload:

```
{
  "iss": "https://registry.example.com",
  "sub": "agent:georoute-planner-v1",
  "_sd_alg": "sha-256",
  "_sd": [
    "9mH4pL5rZ8wQ2nX7cV6sT3kE1oY0jF9uR4vB6nM8fG2"
  ],
  "security": [
    { "google": ["openid", "profile", "email"] },
    { "apiKey": [] }
  ]
}
```

3.2.5. Provider Information

The provider claim contains A2A AgentProvider information about the organization that operates the agent.

3.2.5.1. Original values

The original provider claim in the agent metadata contains:

```
{
  "provider": {
    "organization": "Example Geo Services Inc.",
    "url": "https://www.examplegeoservices.com"
  }
}
```

3.2.5.2. SD-JWT Transformation Process:

When converting the provider claim to selective disclosure, the Agent Registry performs the following transformation:

Step 1: Generate Salt Value

A cryptographically secure random salt is generated for the provider claim:

Salt: "_26bc4LT-ac6q2KI6cBW5es"

Step 2: Create Disclosure Array

The provider claim is packaged into a disclosure array format:

```
[
  "_26bc4LT-ac6q2KI6cBW5es",
  "provider",
  {
    "organization": "Example Geo Services Inc.",
    "url": "https://www.examplegeoservices.com"
  }
]
```

Step 3: Base64url Encode Disclosure

The disclosure array is JSON-serialized and base64url-encoded:

WyJfMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsInByb3ZpZGVyIix7Im9yZ2FuaXphdGlvbiI6IkV4YW1wbGUgR2VvI
FNlcnZpY2VzIEluYy4iLCJlcmwiOiJodHRwciovL3d3dy5leGFtcGxlZ2Vvc2VydmljZXMuY29tInld

Step 4: Calculate SHA-256 Digest

A SHA-256 hash is computed over the base64url-encoded disclosure:

```
digest = SHA256(base64url_encode(disclosure_array))
       = "7kRt2qW5mX9jN4cF7zPSE8nG1vH0oY3lM6xC9fK2bA8"
```

Step 5: Include Digest in SD-JWT

The digest is included in the _sd array of the Issuer-signed JWT payload:

```
{
  "iss": "https://registry.example.com",
  "sub": "agent:georoute-planner-v1",
  "_sd_alg": "sha-256",
  "_sd": [
    "7kRt2qW5mX9jN4cF7zPSE8nG1vH0oY3lM6xC9fK2bA8"
  ]
}
```

```
}
```

3.2.6. Input and Output Modes

The `defaultInputModes` and `defaultOutputModes` claims specify supported content types for A2A message interactions.

3.2.6.1. Original values

The original `defaultInputModes` and `defaultOutputModes` claims in the agent metadata contain:

```
{
  "defaultInputModes": ["application/json", "text/plain"],
  "defaultOutputModes": ["application/json", "image/png"]
}
```

3.2.6.2. SD-JWT Transformation Process:

When converting the `defaultInputModes` and `defaultOutputModes` claims to selective disclosure, the Agent Registry performs the following transformation:

Step 1: Generate Salt Value

A cryptographically secure random salt is generated for the input/output modes claims:

Salt: `"_26bc4LT-ac6q2KI6cBW5es"`

Step 2: Create Disclosure Arrays

Each claim is packaged into separate disclosure array formats:

3.2.6.3. Disclosure for `defaultInputModes`:

```
[
  "_26bc4LT-ac6q2KI6cBW5es",
  "defaultInputModes",
  ["application/json", "text/plain"]
]
```

3.2.6.4. Disclosure for `defaultOutputModes`:

```
[
  "_26bc4LT-ac6q2KI6cBW5es",
  "defaultOutputModes",
  ["application/json", "image/png"]
]
```

Step 3: Base64url Encode Disclosures

The disclosure arrays are JSON-serialized and base64url-encoded:

3.2.6.5. `defaultInputModes` encoded:

`WyJfMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsImRlZmFlbHRJbnBldElvZGVzIixbImFwcGxpY2F0aW9uL2pzbi24iLCJ0ZXh0L3BsYWluIl1d`

3.2.6.6. `defaultOutputModes` encoded:

`WyJfMjZiYzRMVC1hYzZxMktJNmNCVzVlcyIsImRlZmFlbHRPdXRwdXRNb2RlcyIsWyJhcHBsaWNhdGlvbi9qc29uIiwiaWlhZ2ZUvcG5nIl1d`

Step 4: Calculate SHA-256 Digests

SHA-256 hashes are computed over the base64url-encoded disclosures:

```
digest_input = SHA256(base64url_encode(defaultInputModes_disclosure))
               = "5nK3wL8tB4eR9mY2cF6qG7oT1vX0jH3pN4rS8zE5fC6"

digest_output = SHA256(base64url_encode(defaultOutputModes_disclosure))
               = "2pM8fT4nR6oX3cG7zE9qL5wY1vS0jF6kH2nB8mC4rT9"
```

Step 5: Include Digests in SD-JWT

The digests are included in the `_sd` array of the Issuer-signed JWT payload:

```
{
  "iss": "https://registry.example.com",
  "sub": "agent:georoute-planner-v1",
  "_sd_alg": "sha-256",
  "_sd": [
    "5nK3wL8tB4eR9mY2cF6qG7oT1vX0jH3pN4rS8zE5fC6",
    "2pM8fT4nR6oX3cG7zE9qL5wY1vS0jF6kH2nB8mC4rT9"
  ]
}
```

3.2.7. Documentation and Support

Additional A2A Agent Card metadata for documentation and extended capabilities.

```
"iconUrl": "https://georoute-agent.example.com/icon.png",
"documentationUrl": "https://docs.examplegeoservices.com/georoute-agent/api",
"supportsAuthenticatedExtendedCard": true
```

3.3. Agent Card Issuance

Agent Cards are issued by trusted Agent Registries, the details of the which are out of scope of this specification

3.3.1. Issuance Process Overview

1. **Agent Registration:** The agent registers with the Agent Registry, providing its capabilities, endpoints, and authentication information. This includes submitting detailed metadata about the agent's functions, operational requirements, and security capabilities.
2. **Identity Verification:** The Agent Registry verifies the agent's identity and authorization to claim the specified capabilities.
3. **Agent Card Creation:** The Agent Registry creates an SD-JWT Agent Card containing the agent's metadata with appropriate claims marked for selective disclosure. The registry determines which claims should be selectively disclosable based on authorization, privacy policies and security requirements.
4. **Batch Issuance:** Multiple Agent Cards with different disclosure configurations may be issued for different discovery contexts. This enables the same agent to interact in various contexts while maintaining privacy and unlinkability.

Example issuance request based on HTTP POST method with bearing token is shown below:

```
POST /agents/register HTTP/1.1
Host: registry.example.com
Content-Type: application/json
Authorization: Bearer <registry_auth_token>
```



```
{
  "agent_id": "ai-assistant-v2",
  "name": "AI Assistant Agent",
  "description": "General purpose AI assistant",
  "version": "2.1.0",
  "public_key": {
    "kty": "EC",
    "crv": "P-256",
    "x": "TCAER19Zvu3OHF4j4W4vfSVoHIP1ILilDls7vCeGemc",
    "y": "ZxjiWWbZMQGHVWKVQ4hbSIirsVfuecCE6t4jT9F2HZQ"
  },
  "disclosure_contexts": [
    {
      "context": "public",
      "disclose": ["skills", "endpoints", "operational_info"]
    },
    {
      "context": "internal",
      "disclose": ["skills", "endpoints", "operational_info",
        "provider", "security_context"]
    },
    {
      "context": "diagnostic",
      "disclose": ["operational_info", "provider"]
    }
  ]
}
```

4. Agent Discovery Protocols with Privacy Protection

This section describes the protocols and mechanisms for discovering agents:

4.1. Well-Known URI Discovery

The well-known URI discovery mechanism is extended to support SD-JWT Agent Cards:

```
GET /.well-known/agent.json HTTP/1.1
Host: agent.example.com
Accept: application/vnd.sd-jwt+json
```

The application/vnd.sd-jwt+json media type indicates that the client accepts responses containing Selective Disclosure JWT data formatted as JSON. In the examples below, the "agent_card" field contains an SD-JWT formatted according to the Agent Card Structure defined in Section 3.1.

The response contains an SD-JWT Agent Card with context-appropriate disclosures.

```
HTTP/1.1 200 OK
Content-Type: application/vnd.sd-jwt+json
```

```
{
  "agent_card": "eyJhbGciOiJFUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJod...~WyJzYWx0IiwibmFtZSI6IkJFIEFzc2lzdGFudCJd~WyJzYWx0MiIsImVuZHBvaW50cyIsW3sibmFtZSI6InByaWlhcnkiLCJlcmwiOiJodHRwciovL2FnZW50LmV4YW1wbGUuY29tL2FwaS92MSJ9XV0~"
}
```

For privacy-sensitive discovery, the agent MAY require authentication before revealing the Agent Card:

```
GET /.well-known/agent.json HTTP/1.1
Host: agent.example.com
```

Authorization: Bearer <discovery_token>
Accept: application/vnd.sd-jwt+json

4.2. Registry-Based Discovery

Agent Registries support discovery with context-based selective disclosure. In the example below, an agent is discovered based on the skill "route-optimizer-traffic".

```
POST /agents/discover HTTP/1.1
Host: registry.example.com
Content-Type: application/json
Authorization: Bearer <registry_access_token>
```

```
{
  "query": {
    "skills": ["route-optimizer-traffic"],
    "availability": "24/7"
  },
  "context": "public",
  "max_results": 10
}
```

4.3. Contextual Discovery

Discovery can be performed with different contexts to control the level of information disclosure:

Public Context: Minimal information suitable for public discovery, including basic skills and public endpoints.

Internal Context: Additional operational details for internal organizational use.

Diagnostic Context: Detailed technical information for troubleshooting and monitoring.

Federation Context: Inter-organizational discovery with appropriate trust relationships.

The same agent may present different Agent Cards for different contexts while maintaining unlinkability between presentations.

4.4. Integration with Model Context Protocol

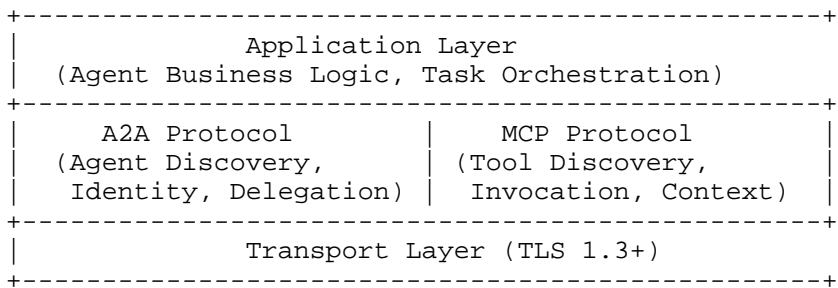
The A2A protocol and SD-Card operate at the agent-to-agent communication layer, while Model Context Protocol (MCP) [MCP-SPEC] addresses agent-to-tool interactions. These protocols are complementary:

- * **A2A/SD-Card**: Agent discovery, identity, and capability negotiation
- * **MCP**: Tool discovery, invocation, and context management

When an agent discovered via SD-Card utilizes MCP for tool access, the following security considerations apply:

1. The SD-Card's security context does not automatically propagate to MCP tool invocations
2. Separate authorization may be required for tool access
3. Audit logs SHOULD correlate A2A interaction IDs with MCP sessions

4.4.1. Protocol Layering



SD-Card enables secure agent discovery and authentication, after which agents may use MCP for accessing tools and external resources. The `interaction_id` from the Key Binding JWT SHOULD be included in MCP requests for audit correlation.

5. Cryptographic Authentication and Capability-Based Authorization

SD-Card provides enhanced authentication and authorization mechanisms through SD-JWT integration with cryptographic key binding. This section details the complete authentication flow, key binding mechanisms, and capability-based authorization processes that ensure secure agent interactions.

5.1. SD-JWT with Key Binding

Selective Disclosure JWT (SD-JWT) with Key Binding is a security mechanism that enables privacy-preserving authentication.

An SD-JWT consists of three main parts separated by tildes (~):

- 1. Issuer-Signed JWT: Contains the base claims and selective disclosure metadata
- 2. Disclosure Arrays: Base64url-encoded arrays containing the salt and claim data
- 3. Key Binding JWT (KB-JWT): Proves possession of the confirmation key

SD-JWT Format:
<Issuer-Signed JWT>~<Disclosure 1>~<Disclosure 2>~...~<Key Binding JWT>

Key binding cryptographically proves that the presenter of an SD-JWT possesses the private key corresponding to the public key specified in the `cnf` (confirmation) claim of the SD-JWT. This mechanism:

- 1. Selective Disclosure: Only relevant claims are revealed to the verifier
- 2. Cryptographic Binding: Proof of possession of a private key
- 3. Unlinkable Presentations: Different interactions cannot be correlated
- 4. Replay Protection: Each presentation is unique and time-bound

5.2. Agent Authentication Process

The agent authentication process involves multiple cryptographic steps to ensure secure and verifiable identity establishment:

5.2.1. Step 1: Agent Card Presentation

The agent presents its SD-JWT Agent Card with context-appropriate

selective disclosures. The "agent_card" field contains an SD-JWT generated from the following raw input structure:

The agent_card field is generated from the following base agent information:

Selective Disclosures:

Resulting SD-JWT Presentation:

5.2.2. Step 2: Key Binding JWT Creation

```
{
  "alg": "ES256",
  "typ": "kb+jwt"
}
```

KB-JWT Payload:

```
{
  "iat": 1704063600,
  "aud": "client.example.com",
  "sd_hash": "Kt7QqNZBzqnJKVQ4hbSIirsVfuecCE6t4jT9F2HZQ",
  "interaction_id": "12345678-1234-1234-1234-123456789abc",
  "nonce": "random-nonce-value-12345"
}
```

where,

- * iat: Issued at time - prevents replay attacks
- * aud: Audience - identifies the intended recipient
- * sd_hash: Hash of the SD-JWT presentation (binds KB-JWT to specific presentation)
- * interaction_id: Unique interaction identifier - prevents cross-session attacks
- * nonce: Optional random value - adds additional replay protection

5.2.3. Step 3: Cryptographic Verification

The client performs comprehensive verification:

1. Agent Card Signature Verification: Validates the issuer's signature
2. Disclosure Verification: Confirms disclosed claims match their digests
3. Key Binding Verification: Validates the KB-JWT signature using the agent's public key
4. Temporal Validation: Checks expiration and freshness of timestamps
5. Audience Validation: Ensures the KB-JWT is intended for this client

5.3. Advanced Authorization Flows

5.3.1. Capability-Based Discovery

Verifiers can query for agents based on required capabilities:

```
POST /agents/discover HTTP/1.1
Host: registry.example.com
Content-Type: application/json
Authorization: Bearer <registry_access_token>
```

```
{
  "query": {
    "skills": ["route-optimizer-traffic"],
    "capabilities": {
      "streaming": true
    }
  },
  "context": "internal",
  "requester_claims": {
    "organization": "example.com",
    "roles": ["service-account"]
  }
}
```

```
}
}
```

5.3.2. Role-Based Disclosure

The Agent Registry MAY implement role-based disclosure policies:

Context	Disclosed Claims
public	name, description, skills (names only)
internal	All public + endpoints, capabilities, provider
diagnostic	All internal + securitySchemes, version details
federation	Context-dependent based on federation agreement

Table 1

5.3.3. Token Exchange for Sub-Agent Delegation

When a client agent needs to delegate to a discovered agent, it MAY use OAuth 2.0 Token Exchange [RFC8693]:

```
POST /token HTTP/1.1
Host: auth.example.com
Content-Type: application/x-www-form-urlencoded
```

```
grant_type=urn:ietf:params:oauth:grant-type:token-exchange
&subject_token=<client_agent_token>
&subject_token_type=urn:ietf:params:oauth:token-type:access_token
&requested_token_type=urn:ietf:params:oauth:token-type:access_token
&audience=agent:georoute-planner-v1
&scope=route-optimizer-traffic
```

The authorization server validates the subject token, applies any scope restrictions, and returns a new token bound to the target agent.

6. IANA Considerations

6.1. Media Type Registration

This specification registers the following media type with IANA:

Type name: application

Subtype name: vnd.sd-jwt+json

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: binary (JSON)

Security considerations: See Security Considerations section of this document

Interoperability considerations: N/A

Published specification: This document

Applications that use this media type: A2A agent discovery systems, agent registries, and agent clients that support selective

disclosure of agent capabilities

Fragment identifier considerations: N/A

6.2. Verifiable Credential Type Registration

This specification registers the following verifiable credential type in the "OAuth URI" registry:

URI: urn:ietf:params:oauth:token-type:sd-agent-card

Description: SD-JWT Agent Card for A2A protocol

Reference: This document

6.3. JSON Web Token Claims Registration

This specification does not define new JWT claims but uses existing claims as defined in [RFC9901] and standard JWT claims from [RFC7519]

7. Security Considerations

7.1. Token Binding and Replay Prevention

SD-Card implementations MUST implement token binding mechanisms to prevent stolen credential misuse. Implementations SHOULD support DPOP [RFC9449] for application-level proof-of-possession when presenting SD-Cards outside the key binding JWT mechanism.

The Key Binding JWT already provides replay protection through:

- * iat (issued-at) timestamps with short validity windows
- * aud (audience) binding to specific verifiers
- * sd_hash binding to specific SD-JWT presentations
- * Optional nonce values for additional freshness

Verifiers MUST reject Key Binding JWTs with:

- * iat timestamps more than 5 minutes in the past
- * Missing or incorrect aud claims
- * Mismatched sd_hash values

7.2. Agent Identity Verification

In multi-agent workflows, identity verification becomes critical. The architecture aligns with WIMSE [WIMSE-ARCH] principles for workload identity:

1. ***Agent Attestation***: The cnf claim provides cryptographic proof that the presenter possesses the corresponding private key
2. ***Issuer Trust***: The Agent Registry's signature establishes a trust anchor. Verifiers MUST validate the issuer's signature using pre-established trust relationships
3. ***Claim Integrity***: Selective disclosure digests ensure that disclosed claims have not been modified since issuance

7.3. Delegation and Multi-Agent Scenarios

When agents delegate tasks to sub-agents, the following security

properties MUST be maintained:

1. ***Delegation Chains***: Use OAuth 2.0 Token Exchange [RFC8693] to create constrained tokens for sub-agents with reduced scope
2. ***Audit Trail***: Each delegation SHOULD include an `interaction_id` that enables tracing across agent invocations
3. ***Scope Limitation***: Sub-agents MUST NOT receive broader permissions than the delegating agent possesses
4. ***Human Oversight***: For sensitive operations, delegation chains SHOULD support human-in-the-loop approval mechanisms

7.4. Privacy Considerations

7.4.1. Correlation Prevention

SD-Card's selective disclosure mechanism prevents unwanted correlation:

- * Different salt values per disclosure create unique hashes
- * Verifiers cannot correlate presentations across contexts without access to the original disclosures
- * Batch issuance enables multiple unlinkable SD-Cards for the same agent

7.4.2. Minimal Disclosure Principle

Following NIST AI RMF [NIST-AI-RMF] principles for trustworthy AI:

- * Agents SHOULD request only the disclosures necessary for the interaction context
- * Discovery contexts (public, internal, diagnostic, federation) enable appropriate disclosure levels
- * Verifiers SHOULD NOT request more claims than required for authorization decisions

7.4.3. Data Minimization in Agent Cards

Agent operators SHOULD:

- * Avoid including PII in selectively disclosable claims when possible
- * Use capability descriptions rather than detailed implementation details
- * Rotate SD-Cards periodically to limit temporal correlation
- * Consider using different SD-Cards for different trust domains

7.5. Transport Security

SD-Card provides application-layer security that complements, but does not replace, transport-layer security:

1. ***TLS Required***: All SD-Card exchanges MUST occur over TLS 1.3 or later as recommended in [RFC9700]
2. ***Defense in Depth***: Key binding provides protection even if transport security is compromised

3. ***Channel Binding***: Implementations MAY bind presentations to the TLS session for additional protection against token export attacks

7.6. Issuer Security

Agent Registries (issuers) have significant security responsibilities:

1. ***Key Protection***: Issuer signing keys MUST be protected using hardware security modules (HSMs) or equivalent protection
2. ***Revocation***: Issuers SHOULD implement revocation mechanisms for compromised SD-Cards. This MAY include:
 - * Short-lived credentials requiring frequent renewal
 - * Status list endpoints for revocation checking
 - * Push-based revocation notifications
3. ***Audit Logging***: Issuers MUST maintain audit logs of all SD-Card issuances including agent identity, disclosed claims, and issuance timestamps

7.7. Cryptographic Agility

This specification mandates support for:

- * ES256 (ECDSA with P-256 and SHA-256) as the minimum required algorithm
- * SHA-256 for selective disclosure hashing

Implementations SHOULD be prepared to support additional algorithms as they become standardized. Post-quantum cryptographic algorithms for JOSE/COSE are under active development and implementations SHOULD plan for migration paths.

{backmatter}

8. References

8.1. Normative References

- [A2A-SPEC] Foundation), A. P. (Linux., "Agent2Agent (A2A) Protocol Specification", Version 1.0, 2025, <<https://a2a-protocol.org/latest/>>.
- [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>>.
- [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>>.
- [RFC9901] Fett, D., Yasuda, K., and B. Campbell, "Selective Disclosure for JWTs (SD-JWT)", RFC 9901, November 2025, <<https://www.rfc-editor.org/rfc/rfc9901.html>>.

8.2. Informative References

- [MCP-SPEC] Foundation), M. P. (Linux., "Model Context Protocol

Specification", 2025, <<https://modelcontextprotocol.io/>>.

[NIST-AI-RMF]

NIST, "AI Risk Management Framework (AI RMF 1.0)", NIST AI 100-1, January 2023,
<<https://www.nist.gov/itl/ai-risk-management-framework>>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012,
<<https://www.rfc-editor.org/rfc/rfc6749>>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <<https://www.rfc-editor.org/rfc/rfc7515>>.

[RFC7519] 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>>.

[RFC8693] "OAuth 2.0 Token Exchange", RFC 8693, January 2020, <<https://www.rfc-editor.org/rfc/rfc8693.html>>.

[RFC9449] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9449, September 2023, <<https://www.rfc-editor.org/rfc/rfc9449.html>>.

[RFC9700] "OAuth 2.0 Security Best Current Practice", RFC 9700, 2024, <<https://www.rfc-editor.org/rfc/rfc9700.html>>.

[WIMSE-ARCH]

Group, W. W., "Workload Identity in a Multi System Environment Architecture", Work in Progress, Internet-Draft, draft-ietf-wimse-arch, 2025, <<https://datatracker.ietf.org/doc/draft-ietf-wimse-arch/>>.

Authors' Addresses

Suhas Nandakumar
Cisco
Email: snandaku@cisco.com

Cullen Jennings
Cisco
Email: flufy@cisco.com