WIMSE
Internet-Draft
Intended status: Standards Track
Expires: 26 September 2026

D. Liu
H. Zhu
J. Jin
Alibaba Group

25 March 2026

Carrying Remote Attestation Evidence in Workload Identity Tokens (WIT)
draft-liu-wimse-wit-attestation-00

Abstract

This document specifies how Remote Attestation evidence, as defined by the IETF RATS architecture, can be conveyed within a Workload Identity Token (WIT) as used in the WIMSE (Workload Identity for Micro-Services Environments) framework. The WIT includes attestation measurements that enable fast-path policy evaluation without requiring immediate access to full evidence. The WIT is bound to the HTTP request using OAuth 2.0 Demonstrating Proof-of-Possession (DPoP), ensuring that attestation claims are protected against replay and token theft.

This specification defines a two-tier verification model: lightweight verification using embedded measurements for common scenarios, and deep verification using externalized evidence for high-assurance requirements. This enables secure, cross-domain verification of workload integrity without requiring direct access to platform-specific reference values, while enabling efficient deployments.

Status of This Memo

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

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

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

This Internet-Draft will expire on 26 September 2026.

Table of Contents

1. Introduction

In some scenarios of the security-sensitive environments, workloads need to prove not only their identity but also the integrity of their execution environment. The WIMSE framework defines the Workload Identity Token (WIT) for conveying identity between workloads. However, to support confidential computing and remote attestation across administrative domains (e.g., multi-cloud), the WIT need to carry verifiable attestation evidence.

This document extends the WIT specification by defining standard claims for attestation and specifying their integration with WIMSE and the RATS architecture [RFC9334].

2. 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.

This document uses terms from the RATS architecture [RFC9334], including "Evidence", "Endorsement", and "Reference Value". It also assumes familiarity with WIMSE [I-D.ietf-wimse-s2s-protocol] and OAuth 2.0 DPoP [RFC9449].

3. Attestation Claims in the WIT

A WIT conveying attestation information MUST include the claims defined in this section. When attested_environment is true, the measurements claim MUST be present to enable fast-path policy evaluation. All attestation claims MUST be protected by the WIT’s digital signature and bound to the request via DPoP. The WIT implementation follows the JWT Best Current Practices [RFC8725].

This specification defines a two-tier verification model:

* *Fast Path*: Relying parties evaluate embedded measurements against local policy without fetching external evidence. This covers the majority of requests with sub-millisecond latency.

* *Deep Path*: For high-assurance scenarios, relying parties retrieve full evidence via evidence_ref and perform cryptographic verification of the complete attestation chain.

3.1. attested_environment

The "attested_environment" claim is a REQUIRED boolean value indicating whether the workload is running in an environment capable of producing valid remote attestation evidence.

If true, the measurements claim MUST be present, and the verifier SHOULD evaluate the measurements against its local policy. If false or absent, no attestation verification is performed.

3.2. tee_type

The "tee_type" claim is a REQUIRED string (when attested_environment is true) identifying the type of Trusted Execution Environment (TEE). This value determines the format of the measurements claim. Registered values include:

* "intel-tdx" - Intel Trust Domain Extensions
* "amd-sev-snp" - AMD Secure Encrypted Virtualization - Secure Nested Paging
* "intel-sgx" - Intel Software Guard Extensions
* "arm-cca" - ARM Confidential Compute Architecture

This value enables policy-based decisions (e.g., "only accept intel-tdx") and determines how the verifier interprets the measurements claim.

New TEE types MAY be registered through the IANA registry defined in Section 7.2.

3.3. measurements

The "measurements" claim is REQUIRED when attested_environment is true. It contains a summary of key attestation measurements that enable fast-path policy evaluation without retrieving full evidence. The measurements object provides sufficient information for relying parties to make authorization decisions for common scenarios.

The measurements object MUST contain the following members:

type: REQUIRED string identifying the measurement format. This value MUST correspond to the tee_type claim. Valid values are defined in Section 3.5.

algorithm: REQUIRED lowercase string specifying the hash algorithm

used for measurements. Valid values are "sha256", "sha384", and "sha512". These identifiers correspond to the IANA Hash Function Textual Names registry. Implementations MUST support "sha384" for TEE measurements; other algorithms MAY be used if specified by the TEE type definition.

registers: REQUIRED object containing TEE-specific measurement registers. The structure of this object depends on the type field and is defined in Section 3.5.

summary: OPTIONAL string containing a hash of all measurements for quick comparison. Format: "algorithm:hex_value" where algorithm is a hash algorithm identifier (e.g., "sha384") and hex_value is the lowercase hexadecimal representation without the "0x" prefix (e.g., "sha384:abc123..."). The algorithm identifier SHOULD match the algorithm used in the measurements claim.

Verifiers MUST validate that the type field matches the tee_type claim. Verifiers MAY use the summary field for fast comparison against known-good values before inspecting individual registers.

3.4. evidence_ref

The "evidence_ref" claim is an OPTIONAL URI pointing to a resource that contains the full RATS Evidence (e.g., in CWT or EAT format). The resource MUST be accessible over HTTPS with TLS 1.2 or higher, and verifiers MUST validate the server certificate according to their trust policy.

This claim enables the deep-path verification tier. When present, relying parties MAY retrieve the full evidence for cryptographic verification of the complete attestation chain. This is RECOMMENDED for high-assurance scenarios or when local policy requires validation beyond the measurements claim.

Example: https://kbs.customer.example/evidence/abc123

The evidence resource MAY be hosted by a customer-operated Key Broker Service (KBS) or cloud provider attestation service. Verifiers MUST authenticate the evidence source and validate signatures according to their trust policy.

Alternatively, full Evidence MAY be embedded directly as a separate claim (e.g., evidence_cwt), but this is NOT RECOMMENDED due to size constraints in HTTP headers. The measurements + evidence_ref approach provides better performance and scalability.

3.5. TEE-Specific Measurement Formats

This section defines the structure of the registers object for each supported TEE type. Implementations MUST follow these formats when generating or validating measurements claims.

3.5.1. Intel TDX Measurements

For tee_type="intel-tdx", the measurements type MUST be "tdx-rtmr" and the registers object MUST contain Runtime Measurement Registers (RTMRs):

```
{
  "type": "tdx-rtmr",
  "algorithm": "sha384",
  "registers": {
    "rtmr0": "hex_encoded_48_bytes",
    "rtmr1": "hex_encoded_48_bytes",
    "rtmr2": "hex_encoded_48_bytes",
    "rtmr3": "hex_encoded_48_bytes"
  },
  "summary": "sha384:abc123..."
}
```

Figure 1

RTMR semantics:

* rtmr0: Initial boot measurements (BIOS, bootloader)

* rtmr1: OS loader and kernel measurements

* rtmr2: System configuration measurements

* rtmr3: Application and runtime measurements

All RTMR values MUST be hex-encoded SHA-384 hashes (96 hex characters). The summary field SHOULD be the SHA-384 hash of the concatenation of rtmr0||rtmr1||rtmr2||rtmr3.

3.5.2. AMD SEV-SNP Measurements

The measurement format for tee_type="amd-sev-snp" is TBD. Future versions of this specification will define the snp-pcr format including vTPM PCR values and SNP-specific launch measurements.

3.5.3. Intel SGX Measurements

The measurement format for tee_type="intel-sgx" is TBD. Future versions of this specification will define the sgx-mr format including MRENCLAVE and MRSIGNER values.

3.5.4. Extensibility

New TEE types can be supported by defining a new measurement type and corresponding registers structure. Implementations encountering an unknown type value MUST either reject the WIT or fall back to evidence_ref verification if present.

To register a new TEE type and measurement format, follow the IANA registration process defined in Section 7.2. The registration MUST include:

* TEE type identifier (e.g., "arm-cca")
* Measurement type identifier (e.g., "cca-rim")
* Hash algorithm(s) used
* Detailed specification of the registers structure
* Semantics of each register field
* Reference to authoritative TEE specification

3.6. Example WIT with Measurements

The following figure shows a complete example of a WIT payload containing attestation measurements:

```
{
  "sub": "spiffe://example.com/ns/default/sa/workload-a",
  "iss": "https://wimse-ca.example.com",
  "iat": 1700000000,
  "exp": 1700003600,
  "jti": "a1b2c3d4-e5f6-7890-g1h2-i3j4k5l6m7n8",

  "attested_environment": true,
  "tee_type": "intel-tdx",

  "measurements": {
    "type": "tdx-rtmr",
    "algorithm": "sha384",
    "registers": {
      "rtmr0": "a1b2c3d4e5f67890123456789012345678901234567890abcdef1234567890abcdef123456789012345
678901234567890abcd",
      "rtmr1": "f1e2d3c4b5a67890123456789012345678901234567890abcdef1234567890abcdef12345
6789012345678abcd",
      "rtmr2": "1a2b3c4d5e6f78901234567890123456789012345678901234567890abcdef1234567890a
bcdef1234567890ab",
      "rtmr3": "9f8e7d6c5b4a32109876543210987654321098765432109876543210fedcba9876543210fedcba9876543210fedcb
a987654321098ab"
    },
    "summary": "sha384:2f5d8c9e1a3b7f4e6d8c2a1b9e7f3d5c8a4b6e1f9d3c7a5b2e8f4d1c6a9b3e7f"
  },

  "evidence_ref": "https://kbs.customer.example/evidence/tdx/abcd1234"
}
```


Figure 2: Example WIT Payload with Attestation Claims

This JSON represents the payload (claims set) of a signed WIT JWT. The attestation claims (attested_environment, tee_type, measurements, evidence_ref) are standard top-level members of the JWT payload. The measurements enable fast-path policy evaluation, while evidence_ref provides access to full attestation evidence for deep verification when required.

The entire WIT is base64url-encoded and signed (e.g., using RS256), forming the token transmitted in the Workload-Identity-Token header. A typical WIT with embedded measurements is approximately 800-1200 bytes when encoded, suitable for HTTP headers.

4. Integration with OAuth 2.0 DPoP

As required by [I-D.ietf-wimse-s2s-protocol], the WIT MUST be bound to the HTTP request using OAuth 2.0 DPoP [RFC9449].

Note that the DPoP proof binds to the exact WIT string via the "ath" claim, which contains the SHA-256 hash of the WIT presented in the Workload-Identity-Token header, ensuring that the attestation claims cannot be substituted or replayed.

The DPoP private key is assumed to be securely held by the workload; this specification does not define how the relying party verifies that the DPoP key was generated within the attested TEE. This verification is delegated to the WIT issuer, which validates the TEE evidence before issuing the WIT.

```
POST /api/data HTTP/1.1
Host: service-b.example
Workload-Identity-Token: eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.xxxxx
DPoP: eyJhbGciOiJFUzM4NCIsInR5cCI6ImRwb3AifQ.yyyyy
```

Figure 3: Example HTTP Request Headers

5. Cross-Domain Attestation Workflow

The following diagram illustrates how a workload in Domain A (e.g., Cloud Provider X) authenticates to a service in Domain B (e.g., Cloud Provider Y) using a WIT that carries attestation measurements. The two-tier verification model enables both fast-path policy evaluation and optional deep verification, with DPoP ensuring request binding and replay protection.

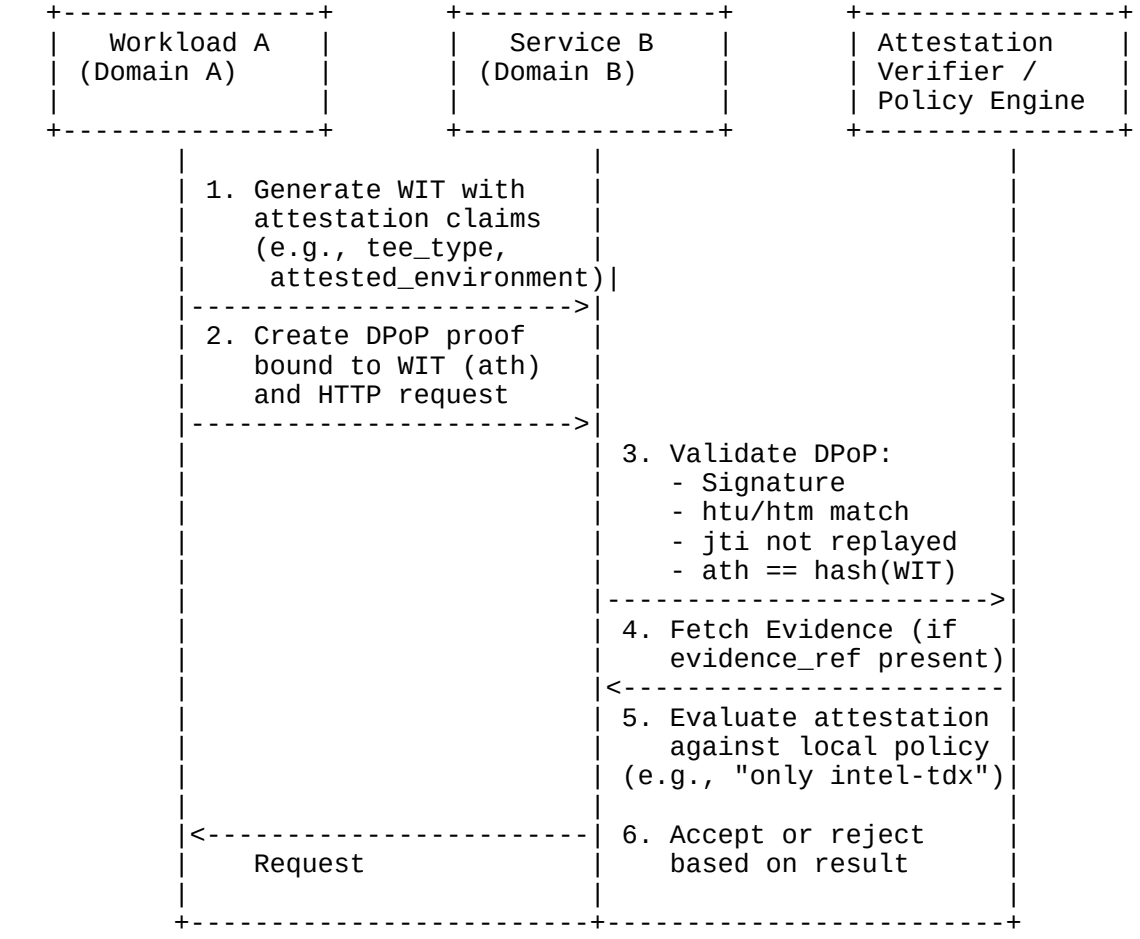

Figure 4: Cross-Domain WIT + Attestation + DPoP Flow

Steps in detail:

1. *WIT Generation*: Workload A constructs a WIT containing identity (e.g., SPIFFE ID) and attestation claims (attested_environment, tee_type, measurements, optionally evidence_ref). The measurements claim contains TEE-specific register values that enable fast policy evaluation.

2. *DPoP Binding*: Workload A generates a DPoP proof that includes the SHA-256 hash of the WIT in the ath claim, along with the HTTP method (htm) and URL (htu). This binds the attestation measurements to this specific request.

3. *Request Transmission*: The client sends an HTTP request with both Workload-Identity-Token and DPoP headers.

4. *DPoP Validation*: Service B validates the DPoP proof per [RFC9449], ensuring the WIT is bound to this specific request and has not been replayed.

5. *Fast-Path Verification*: Service B extracts the measurements claim from the WIT and evaluates it against local policy. For most requests, this step is sufficient and completes efficiently. Examples include checking if tee_type matches policy, comparing measurements.summary against known-good values, verifying specific registers match approved configurations, and checking if measurements are in a revocation list. If the measurements satisfy policy, Service B grants access immediately.

6. *Deep-Path Verification* (optional): For high-assurance scenarios or when fast-path policy is inconclusive, Service B MAY fetch the full RATS Evidence from evidence_ref (over TLS) and perform cryptographic verification of the complete attestation chain. This includes validating the TEE hardware signature, verifying the certificate chain to root-of-trust, checking TCB (Trusted Computing Base) status, and validating measurements match the evidence. Service B MAY cache the evidence verification result to avoid repeated fetches for subsequent requests from the same workload.

7. *Policy Evaluation*: Service B's policy engine makes the final authorization decision based on the verification tier used. The policy may specify different requirements based on operation sensitivity (e.g., read vs. write operations).

Access Decision: Based on the evaluation, Service B grants or denies access. The verification tier used (fast-path vs. deep-path) MAY be logged for audit purposes.

Note that no direct trust relationship between the TEE attestation root (e.g., Intel TDX root key) and Domain B is required. Instead, Domain B relies on its configured policy about which TEE types it accepts.

6. Security Considerations

6.1. Threat Model

This specification assumes the following trust model and threat environment:

Trust Assumptions:

* The WIT issuer (e.g., workload identity provider) is trusted to correctly issue tokens and validate the TEE attestation evidence before inclusion in the WIT. The relying party trusts that the issuer has performed this verification correctly; this specification does not define mechanisms for cryptographically binding the DPoP key to the TEE measurements.

* The TEE hardware root of trust is assumed to be uncompromised for the specific TEE types accepted by the relying party.

* The relying party maintains a secure policy configuration that defines acceptable TEE types and measurement values.

Future versions of this specification may define mechanisms for cryptographically binding the DPoP key to the TEE measurements, enabling relying parties to verify this binding independently of the issuer.

Attacker Model:

* *Network attacker:* Can intercept, modify, or replay network traffic between workloads, but does not possess valid WITs or DPoP keys.

* *Compromised workload:* A workload running outside a TEE attempting to present falsified attestation claims.

* *Malicious relying party:* A service that attempts to harvest or replay attestation evidence for other purposes (mitigated by DPoP binding).

Out of Scope: This specification does not address attacks against the TEE hardware itself (e.g., side-channel attacks, physical attacks), nor does it address compromise of the WIT issuer’s signing keys. Protection against such threats requires additional operational and architectural measures outside the scope of this document.

6.2. Security Properties

The following security considerations apply to implementations of this specification:

Replay Protection: DPoP’s use of jti and request binding (htm, htu,

ath) prevents replay of WITs containing attestation measurements. The ath claim cryptographically binds the measurements to this specific request, preventing an attacker from reusing measurements from a compromised token.

Measurements Integrity: The measurements claim MUST be protected by the WIT's digital signature. Verifiers MUST validate the WIT signature before trusting measurements. The measurements themselves are not sufficient for security without signature verification.

Fast-Path vs. Deep-Path Trade-offs: Fast-path verification using measurements provides performance benefits but relies on local policy rather than cryptographic verification. Deployments MUST assess the risk level of each operation and choose the appropriate verification tier. High-value operations (e.g., financial transactions, sensitive data access) SHOULD use deep-path verification with evidence_ref.

Measurements Freshness: Measurements represent the state of the workload at WIT issuance time. Verifiers SHOULD enforce short WIT lifetimes (exp claim) to ensure measurements remain fresh. For long-running operations, verifiers MAY require periodic re-attestation.

TEE-Specific Vulnerabilities: Different TEE types have different security properties and known vulnerabilities. Verifiers MUST maintain up-to-date knowledge of TEE security status and update their policies accordingly. The measurements alone do not protect against vulnerabilities in the underlying TEE implementation.

Privacy: Attestation measurements may reveal platform details (firmware versions, configuration, etc.). Deployments SHOULD minimize disclosed information. The measurements claim contains only essential values; full evidence via evidence_ref MAY contain more detailed information that should be protected accordingly.

Verifier Policy: The relying party MUST validate attestation measurements against a well-defined local policy. Simply checking that measurements are present is insufficient. Policies SHOULD specify acceptable TEE types, approved measurement values or ranges, requirements for specific registers, and when deep-path verification is required. Trust in the WIT issuer (e.g., cloud provider CA) is assumed but MUST be explicitly configured.

Evidence Source Authentication: When using evidence_ref for deep-

path verification, verifiers MUST authenticate the evidence source and validate that it is authorized to provide evidence for the claimed TEE type. Customer-operated KBS deployments SHOULD use mTLS or equivalent authentication.

7. IANA Considerations

7.1. JSON Web Token Claims Registration

This document registers new claims in the "JSON Web Token Claims" registry established by [RFC7519].

Claim Name: "attested_environment"

Claim Description: Indicates attested execution environment

Change Controller: IETF

Specification Document(s): [[THIS DOCUMENT]], Section 3.1

Claim Name: "tee_type"

Claim Description: Identifies Trusted Execution Environment type

Change Controller: IETF

Specification Document(s): [[THIS DOCUMENT]], Section 3.2

Claim Name: "measurements"

Claim Description: Contains attestation measurements for fast verification

Change Controller: IETF

Specification Document(s): [[THIS DOCUMENT]], Section 3.3

Claim Name: "evidence_ref"

Claim Description: URI reference to full RATS Evidence

Change Controller: IETF

Specification Document(s): [[THIS DOCUMENT]], Section 3.4

7.2. WIMSE TEE Types Registry

IANA is requested to create a new registry titled "WIMSE WIT TEE Types" under the "WIMSE" namespace. This registry maintains identifiers for Trusted Execution Environment types used in the tee_type claim.

7.2.1. Registration Procedure

Registration of new TEE type values follows the "Specification Required" policy as defined in [RFC8126]. The designated experts are appointed by the IETF Security Area Directors. The designated expert SHALL verify that:

* The TEE type is documented in a publicly accessible specification
* The corresponding measurement format is clearly defined
* The TEE type identifier follows the naming convention (lowercase ASCII, hyphen-separated, e.g., "vendor-product")
* The registration does not duplicate an existing entry
* The specification defines security considerations specific to the TEE type

In case of dispute or disagreement with the designated expert’s decision, the registrant may appeal to the IETF Security Area Directors.

The registration template consists of the following fields:

TEE Type: The requested identifier (e.g., "intel-tdx")

Description: A brief description of the Trusted Execution Environment

Measurement Type: The corresponding measurement format identifier

Reference: URI to the specification document

Change Controller: The entity responsible for the registration (e.g., "IETF" or organization name)

7.2.2. Initial Registry Contents

The initial registry contains the following entries:

TEE Type	Description	Measurement Type	Reference
intel-tdx	Intel Trust Domain Extensions	tdx-rtmr	[[THIS DOCUMENT]], Section 3.5.1
amd-sev-snp	AMD Secure Encrypted Virtualization - SNP	TBD	[[THIS DOCUMENT]], Section 3.5.2
intel-sgx	Intel Software Guard Extensions	TBD	[[THIS DOCUMENT]], Section 3.5.3
arm-cca	ARM Confidential Compute Architecture	TBD	TBD

Table 1

8. References

8.1. Normative References

[I-D.ietf-wimse-s2s-protocol]
Authors, W., "Service-to-Service Authentication Using Workload Identity", Work in Progress, Internet-Draft, draft-ietf-wimse-s2s-protocol-07, 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-s2s-protocol-07>.

[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/info/rfc2119>.

[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/info/rfc7519>.

[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/info/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/info/rfc8174>.

[RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://www.rfc-editor.org/info/rfc8725>.

[RFC9334] Birkholz, H., Thaler, D., Richardson, M., Smith, N., and W. Pan, "Remote ATtestation procedureS (RATS) Architecture", RFC 9334, DOI 10.17487/RFC9334, January 2023, <https://www.rfc-editor.org/info/rfc9334>.

[RFC9449] Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)", RFC 9449, DOI 10.17487/RFC9449, September 2023, <https://www.rfc-editor.org/info/rfc9449>.

Authors' Addresses

Dapeng Liu
Alibaba Group
Email: max.ldp@alibaba-inc.com

Hongru Zhu
Alibaba Group
Email: hongru.zhr@alibaba-inc.com

Jian Jin
Alibaba Group
Email: jinjian.jj@alibaba-inc.com