Workload Identity in Multi System Environments
Internet-Draft
Intended status: Standards Track
Expires: 12 July 2026

J. Salowey
CyberArk
Y. Sheffer
Intuit

8 January 2026

WIMSE Workload-to-Workload Authentication with HTTP Signatures
draft-ietf-wimse-http-signature-01

Abstract

The WIMSE architecture defines authentication and authorization for software workloads in a variety of runtime environments, from the most basic ones to complex multi-service, multi-cloud, multi-tenant deployments. This document defines one of the mechanisms to provide workload authentication, using HTTP Signatures. While only applicable to HTTP traffic, the protocol provides end-to-end protection of requests (and optionally, responses), even when service traffic is not end-to-end encrypted, that is, when TLS proxies and load balancers are used. Authentication is based on the Workload Identity Token (WIT).

About This Document

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

The latest revision of this draft can be found at https://ietf-wg-wimse.github.io/draft-ietf-wimse-s2s-protocol/draft-ietf-wimse-s2s-protocol.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-wimse-http-signature/.

Discussion of this document takes place on the Workload Identity in Multi System Environments Working Group mailing list (mailto:wimse@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/wimse/. Subscribe at https://www.ietf.org/mailman/listinfo/wimse/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-wg-wimse/draft-ietf-wimse-s2s-protocol.

Status of This Memo

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

Copyright Notice

Table of Contents

1. Introduction

This document defines authentication and authorization in the context of interaction between two workloads. This is the core component of the WIMSE architecture [I-D.ietf-wimse-arch]. This document focuses on HTTP-based services, and the workload-to-workload call consists of a single HTTP request and its response.

One option to protect such traffic is through Mutual TLS, and this usage is defined in [I-D.ietf-wimse-mutual-tls]. Many deployments prefer application-level approaches, whether for lack of CA infrastructure or because inter-service communication consists of multiple separate TLS hops. This document defines one of the two WIMSE approaches for application-level protection.

We define a profile of the HTTP Signatures protocol [RFC9421] to protect the service traffic. Service authentication uses the Workload Identity Token (WIT) defined in [I-D.ietf-wimse-workload-creds], and the signature uses the private key associated with the WIT and thus proves possession of that key.

As noted, the WIMSE working group is specifying two alternatives for application-level protection, both using the newly introduced Workload Identity Token [I-D.ietf-wimse-workload-creds]. The first alternative [I-D.ietf-wimse-wpt] is inspired by the OAuth DPoP specification. The second is based on the HTTP Message Signatures RFC, and this is the one defined in this document. Appendix B includes a comparison of the two alternatives.

1.1. Deployment Architecture and Message Flow

Refer to Sec. 1.2 of [I-D.ietf-wimse-workload-creds] for the deployment architecture which is common to all three protection options, including the one described here.

2. Conventions and Definitions

All terminology in this document follows [I-D.ietf-wimse-arch].

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.

3. The Protocol: Authentication Based on HTTP Message Signatures

This protocol uses the Workload Identity Token [I-D.ietf-wimse-workload-creds] and the private key associated with its public key, to sign the request and optionally, the response. Formally, this is a profile of the Message Signatures specification [RFC9421].

The request is signed as per [RFC9421]. The following derived components MUST be signed:

* @method
* @request-target

In addition, the following request headers MUST be signed when they exist:

* Content-Type
* Content-Digest
* Authorization
* Txn-Token [I-D.ietf-oauth-transaction-tokens]
* Workload-Identity-Token

If the response is signed, the following components MUST be signed:

* @status

* @method;req
* @request-target;req
* Content-Type if it exists
* Content-Digest if it exists
* Workload-Identity-Token

To ensure the message is fully integrity-protected, if the request or response includes a message body, the sender MUST include (and the receiver MUST verify) a Content-Digest header.

For both requests and responses, the following signature parameters MUST be included:

* created
* expires - expiration MUST be short, e.g. on the order of minutes. The WIMSE architecture will provide separate mechanisms in support of long-lived compute processes.
* nonce
* tag - the value for implementations of this specification is wimse-workload-to-workload

The following signature parameters in the Signature-Input header MUST NOT be used:

* keyid - The signing key is sent along with the message in the WIT. Additionally specifying the key identity would add confusion.
* alg - The signature algorithm is specified in the jwk section of the cnf claim in the WIT. See [I-D.ietf-wimse-workload-creds] and Sec. 3.3.7 of [RFC9421] for details.

It is RECOMMENDED to include only one signature with the HTTP message. If multiple ones are included, then the signature label included in both the Signature-Input and Signature headers SHOULD be wimse.

A sender MUST ensure that each nonce it generates is unique, at least among messages sent to the same recipient. To detect message replays, a recipient SHOULD reject a message (request or response) if a nonce generated by a certain peer is seen more than once.

For clarity: the signature's lifetime (the expires signature parameter) is different and typically much shorter than the WIT's lifetime, denoted by its exp claim.

Implementors need to be aware that the WIT is extracted from the message before the message signature is validated. Recipients of signed HTTP messages MUST validate the signature and content of the WIT before validating the HTTP message signature. They MUST ensure that the message is not processed further before it has been fully validated.

3.1. Signing the Response

Protecting the response by signing it with the server's WIT is RECOMMENDED but optional. In particular, if the response may be exceptionally large or is expected to be streamed, signing it may not be practical.

In practice, we expect response signing to be enabled by local policy. If response signing is enabled for a deployment, the client (recipient of the response) MUST check that the signature exists and validate it. The response MUST be rejected if a signature is absent or fails to validate.

As described in Section 5 of [RFC9421], either client or server MAY send an Accept-Signature header, but is not required to do so. The Accept-Signature header indicates a preference for signed messages but does not mandate that responses be signed. When a client sends Accept-Signature in a request, it SHOULD list the response components it wishes to have signed (including at least those specified above for signed responses). When a server sends Accept-Signature in a response, it SHOULD list the request components it wishes to have signed in subsequent requests (minimally those specified above for signed requests).

3.2. Error Conditions

Errors may occur during the processing of the message signature. If the signature verification fails for any reason, such as an invalid signature, an expired validity time window, or a malformed data structure, an error is returned. Typically, this will be in response to an API call. An HTTP status code such as 400 (Bad Request) is appropriate. The response could include more details as per [RFC9457], such as an indicator that the wrong key material or algorithm was used. The use of HTTP status code 401 is NOT RECOMMENDED for this purpose because it requires a WWW-Authenticate with acceptable http auth mechanisms in the error response and an associated Authorization header in the subsequent request. The use

of these headers for the WIT is not compatible with this specification.

3.3. Example Requests and Responses

Following is a non-normative example of a signed request and a signed response.

The caller uses this keypair:

```
{
  "alg": "EdDSA",
  "crv": "Ed25519",
  "d": "JMHQzsQ7wxHfaj5d4fQJ8oDNGh5SJY1CcOD24tuo2ws",
  "kid": "svc-a-key",
  "kty": "OKP",
  "x": "zKORkA6iFB3-dkDJG6kbnhw-57VXOHa-IE9XWVd-GtM"
}
```

Figure 1: Caller Private Key

The caller uses its keypair and generates the following HTTP request:

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

GET /gimme-ice-cream?flavor=vanilla HTTP/1.1
Host: example.com
Signature: wimse=:b1kQ7vFYUShd9QS82ojrPAy2hAgiIqSED20bXXjwH6xsnXHF0r\
b2J8OeIdbtSupQUsez8IOqQvoYGPaWKu76Cg==:
Signature-Input: wimse=("@method" "@request-target" "workload-identi\
ty-token");created=1761859807;expires=1761860107;nonce="abcd1111";ta\
g="wimse-workload-to-workload"
Workload-Identity-Token: eyJhbGciOiJFZERTQSIsImtpZCI6Imlzc3Vlci1rZXk\
iLCJ0eXAiOiJ3aXQrand0In0.eyJjbmYiOnsiandrIjp7ImFsZyI6IkVkRFNBIiwiY3J\
2IjoiRWQyNTUxOSIsImtpZCI6InN2Yy1hLWtleSIsImt0eSI6Ik9LUCIsIngiOiJ6S09\
Sa0E2aUZCMy1ka0RKRzZrYm5ody01N1ZYT0hhLUlFOVhXVmQtR3RNIn19LCJleHAiOjE\
3NjE4NjAxMDcsImlhdCI6MTc2MTg1OTgwNywiaXNzIjoiaHR0cHM6Ly9leGFtcGxlLmN\
vbS9pc3N1ZXIiLCJqdGkiOiJ3aXQtMTc2MTg1OTgwNzM0NTQ3ODAwMCIsInN1YiI6Ind\
pbXNlOi8vZXhhbXBsZS5jb20vc3ZjQSJ9.KxXGBK2YWhy8XT1tB9Z9sZscn2fb4OHDNz\
2S8mhP8fGK0mCw6yK4lv9wRqO3hnAXWJEzWuWRHpVQO8_82t_IBA
```

Figure 2: Signed Request

Assuming that the workload being called has the following keypair:

```
{
  "alg": "EdDSA",
  "crv": "Ed25519",
  "d": "u1DH-EPbCxpzi1LLJHtsLGZ6_4lvKsUgFHRNYOo5Mx8",
  "kid": "svc-b-key",
  "kty": "OKP",
  "x": "8zj0S94HAaaQ0kKPhts0UdothtSTjZDX7jYC5zdLsX4"
}
```

Figure 3: Callee Private Key

A signed response would be:

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

HTTP/1.1 404 Not Found
Connection: close
Content-Digest: sha-256=:47DEQpj8HBSa+/TImW+5JCeuQeRkm5NMpJWZG3hSuFU\
=:
Content-Type: text/plain
Signature: wimse=:cQiWDdhftD/qYu22pMUxvdqHPxo7IjOaTQ54UxZ5nvXq6Yj7Mv\
avAW8sGJjXNlPXwqvBc1vy0wtOvS6Q5zdVDQ==:
Signature-Input: wimse=("@status" "workload-identity-token" "content\
-type" "content-digest" "@method";req "@request-target";req);created\
=1761859807;expires=1761860109;nonce="abcd2222";tag="wimse-workload-\
to-workload"
Workload-Identity-Token: eyJhbGciOiJFZERTQSIsImtpZCI6Imlzc3Vlci1rZXk\
iLCJ0eXAiOiJ3aXQrand0In0.eyJjbmYiOnsiandrIjp7ImFsZyI6IkVkRFNBIiwiY3J\
2IjoiRWQyNTUxOSIsImtpZCI6InN2Yy1iLWtleSIsImt0eSI6Ik9LUCIsIngiOiI4emo\
wUzk0SEFhYVEwa0tQaHRzMFVkb3RodFNUalpEWDdqWUM1emRMc1g0In19LCJleHAiOjE\
3NjE4NjAxMDksImlhdCI6MTc2MTg1OTgwOSwiaXNzIjoiaHR0cHM6Ly9leGFtcGxlLmN\
vbS9pc3N1ZXIiLCJqdGkiOiJ3aXQtMTc2MTg1OTgwNzM0NTcwMjAwMCIsInN1YiI6Ind\
pbXNlOi8vZXhhbXBsZS5jb20vc3ZjQiJ9.1Z10zR2hYWXuVwBeJDx965agbLf_V8o2Ac\
SKiyayqGQMAEZNyRm1wrSvDmm6Ha6EJW3-1rhF0C10R-usSUYtAQ

No ice cream today.
```

Figure 4: Signed Response

4. Implementation Status

```
// Note to RFC Editor: please remove this section, as well as the
// reference to RFC 7942, before publication.
```

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942].

The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

4.1. Cofide

* Organization: Cofide
* Implementation: https://github.com/cofide/wimse-s2s-httpsig-poc (https://github.com/cofide/wimse-s2s-httpsig-poc)
* Maturity:
  - WIT + HTTP Message Signatures: proof-of-concept
* Coverage: WIT, HTTP Message Signatures
* License: Apache 2.0
* Contact: jason@cofide.io
* Last updated: 13-Nov-2025

5. Security Considerations

This section includes security considerations that are specific to the HTTP Signature protocol defined here. Refer to [I-D.ietf-wimse-workload-creds] for more generic security considerations associated with the workload identity and its WIT representation.

5.1. Workload Identity Token and Proof of Possession

The Workload Identity Token (WIT) is bound to a secret cryptographic key and is always presented with a proof of possession as described in [I-D.ietf-wimse-workload-creds]. The WIT is a general purpose token that can be presented in multiple contexts. The WIT and its PoP are only used in the application-level options, and both are not used in MTLS. The WIT MUST NOT be used as a bearer token. While this helps reduce the sensitivity of the token it is still possible that a token and its proof of possession may be captured and replayed within the PoP's lifetime.

The HTTP Signature profile presented here binds the proof of possession to the critical parts of the HTTP request (and potentially response), including the Request URI and the message body. This eliminates most of the risk associated with active attackers on a middlebox.

In addition, the following mitigations should be used:

* Preventing Eavesdropping and Interception with TLS

An attacker observing or intercepting the communication channel can view the token and its proof of possession and attempt to replay it to gain an advantage. In order to prevent this the token and proof of possession MUST be sent over a secure, server authenticated TLS connection unless a secure channel is provided by some other mechanisms. Hostname validation according to Section 6.3 of [RFC9525] MUST be performed by the client.

* Limiting Signature Lifespan

The signature lifespan MUST be limited by using a tight expires value, taking into account potential clock skew and processing latency, but usually within minutes of the message sending time. Signatures received outside their validity time MUST be rejected.

* Replay Protection

A signed message includes the jti claim that MUST uniquely identify it, within the scope of a particular sender. This claim SHOULD be used by the receiver to perform basic replay protection against messages it has already seen. Depending upon the design of the system it may be difficult to synchronize the replay cache across all messages validators. If an attacker can somehow influence the identity of the validator (e.g. which cluster member receives the message) then replay protection would not be effective.

5.2. Middle Boxes

In some deployments the Workload Identity Token and proof of possession (signature) may pass through multiple systems. The communication between the systems is over TLS, but the WIT and signature are available in the clear at each intermediary. While the intermediary cannot modify the token or the information within the signature they can attempt to capture and replay the the message or modify unsigned information, such as proprietary HTTP headers that may remain unsigned.

Mitigations listed in the protocol provide a reasonable level of security in these situations, in particular if responses are signed in addition to requests.

5.3. Privacy Considerations

WITs and the signatures may contain private information such as user names or other identities. Care must be taken to prevent disclosure of this information. The use of TLS helps protect the privacy of WITs and proofs of possession.

WITs are typically associated with a workload and not a specific user, however in some deployments the workload may be associated directly to a user. While these are exceptional cases a deployment should evaluate if the disclosure of WITs or signatures can be used to track a user.

6. Security Goals

This section defines semiformal security goals for this protocol, when used in conjunction with the WIT credential. Our aim is to inform developers and for these goals to eventually evolve into formal verification of the protocol.

6.1. Prerequisites

The following are out of scope of the protocol and their security is assumed.

* There exists a WIT Issuer which is trusted to issue credentials honestly.

* Workloads have a way to authenticate themselves to the Issuer and be provisioned with a valid WIT, associated with their WIMSE identity.

* All workloads are provisioned with trust anchors that allow them to validate incoming WITs.

* The entire authorization subsystem is out of scope and trusted. This can potentially include provisioning and enforcement of an authorization policy, issuance of transactions tokens and workload attestation.

* All workload-to-workload traffic is TLS-protected. However TLS may be terminated on one or more middleboxes and the TLS endpoint identity (or identities) is not associated with a WIMSE identity.

* As a result, all workload-to-workload traffic is confidential and (assuming honest participants) is only available to sender, receiver, and any TLS-terminating middleboxes that process the traffic.

6.2. Authentication

* A workload receiving a request can validate that it is signed correctly, and can identify the sender.

* A workload receiving a response can similarly authenticate its sender, provided that optional response signing has been activated and likewise, the recipient validates this signature.

* The above implies that a stolen WIT cannot be used by an entity other than its owner.

6.3. Integrity

* No requests can be modified without detection by the recipient. Integrity of all present HTTP headers specified in this document is protected, as well as the message body (when present).

* No responses can be modified without detection, provided that optional response signing has been activated and that the recipient validates incoming responses.

* Note: Headers not specified in this document may remain unsigned and could potentially be modified or deleted by intermediaries without detection.

6.4. Replay and Deletion

* Replay protection is not strictly mandated because of implementation considerations (e.g., distributed system challenges with synchronizing replay caches across validators). Therefore it is not claimed as a goal, though implementations SHOULD attempt to detect replays where feasible. We note that since most of the message is signed, replay attacks are only possible in a context where the request would be accepted as valid, and this mitigates the risk to some extent.

* Unless response signing is mandated by local policy, complete deletion of a request/response pair is possible without detection.

7. IANA Considerations

This document does not include any IANA considerations.

8. References

8.1. Normative References

[I-D.ietf-wimse-workload-creds]
Campbell, B., Salowey, J. A., Schwenkschuster, A., Sheffer, Y., and Y. Rosomakho, "WIMSE Workload Credentials", Work in Progress, Internet-Draft, draft-ietf-wimse-workload-creds-00, 3 November 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-workload-creds-00>.

[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>.

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

[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>.

[RFC9421] Backman, A., Ed., Richer, J., Ed., and M. Sporny, "HTTP Message Signatures", RFC 9421, DOI 10.17487/RFC9421, February 2024, <https://www.rfc-editor.org/rfc/rfc9421>.

[RFC9525] Saint-Andre, P. and R. Salz, "Service Identity in TLS", RFC 9525, DOI 10.17487/RFC9525, November 2023, <https://www.rfc-editor.org/rfc/rfc9525>.

8.2. Informative References

[I-D.ietf-oauth-transaction-tokens]
Tulshibagwale, A., Fletcher, G., and P. Kasselman, "Transaction Tokens", Work in Progress, Internet-Draft, draft-ietf-oauth-transaction-tokens-06, 28 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-oauth-transaction-tokens-06>.

[I-D.ietf-wimse-arch]
Salowey, J. A., Rosomakho, Y., and H. Tschofenig, "Workload Identity in a Multi System Environment (WIMSE) Architecture", Work in Progress, Internet-Draft, draft-ietf-wimse-arch-06, 30 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-arch-06>.

[I-D.ietf-wimse-mutual-tls]
Salowey, J. A. and Y. Rosomakho, "Workload Authentication Using Mutual TLS", Work in Progress, Internet-Draft, draft-ietf-wimse-mutual-tls-00, 2 November 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-mutual-tls-00>.

[I-D.ietf-wimse-wpt]
Campbell, B. and A. Schwenkschuster, "WIMSE Workload Proof Token", Work in Progress, Internet-Draft, draft-ietf-wimse-wpt-00, 3 November 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-wimse-wpt-00>.

[RFC9457] Nottingham, M., Wilde, E., and S. Dalal, "Problem Details for HTTP APIs", RFC 9457, DOI 10.17487/RFC9457, July 2023, <https://www.rfc-editor.org/rfc/rfc9457>.

Appendix A. Document History

// RFC Editor: please remove before publication.

A.1. draft-ietf-wimse-http-signature-01

* Clarified response signing.

* Clarified signature vs. token lifetime.
* Added security goals.
* Added an Implementation Status section.

A.2. draft-ietf-wimse-http-signature-00

* Initial version, extracted from the -07 draft with minimal edits.

A.3. draft-ietf-wimse-s2s-protocol-07

* Rework the WPT’s oth claim.
* Update the media types.
* Discuss extensibility of WIT and WPT.
* Clarify error handling, specifically why not HTTP 401.
* Correct the code examples.
* Add registration request content for a wimse URI scheme.
* New section on key management.
* Use of the Accept-Signature header.

A.4. draft-ietf-wimse-s2s-protocol-06

* Explicit definition of the Workload Identity Certificate.
* Definition of the validation of workload identifiers as part of workload authentication. Still work in progress.

A.5. draft-ietf-wimse-s2s-protocol-05

* Removed the entire Workload Identity section which is now covered in the Architecture document.
* Content-Digest is mandatory with HTTP-Sig.
* Some wording on extending the protocol beyond HTTP.
* IANA considerations.

A.6. draft-ietf-wimse-s2s-protocol-04

* Require cnf.jwk.alg in WIT which restricts signature algorithm of WPT or HTTP-Sig.
* Replay protection as a SHOULD for both WPT and HTTP-Sig.
* Consolidate terminology with the Architecture draft.

A.7. draft-ietf-wimse-s2s-protocol-03

* Consistently use "workload".
* Implement comments from the SPIFFE community.
* Make iss claim in WIT optional and add wording about its relation to key distribution.
* Remove iss claim from WPT.
* Make jti claim in WIT optional.
* Error handling for the application level methods.

A.8. draft-ietf-wimse-s2s-protocol-02

* Coexistence with bearer tokens.
* Improve the architecture diagram.
* Some more ABNF.
* Clarified identifiers and URIs.
* Moved an author to acknowledgments.

A.9. draft-ietf-wimse-s2s-protocol-01

* Addressed multiple comments from Pieter.
* Clarified WIMSE identity concepts, specifically "trust domain" and "workload identifier".
* Much more detail around mTLS, including some normative language.
* WIT (the identity token) is now included in the WPT proof of possession.

* Added a section comparing the DPoP-inspired app-level security option to the Message Signature-based alternative.

A.10. draft-ietf-wimse-s2s-protocol-00

* Initial WG draft, an exact copy of draft-sheffer-wimse-s2s-protocol-00

* Added this document history section

Appendix B. Comparing the DPoP Inspired Option with Message Signatures

The two workload protection options have different strengths and weaknesses regarding implementation complexity, extensibility, and security. Here is a summary of the main differences between [I-D.ietf-wimse-wpt] and Section 3.

* The DPoP-inspired solution is less HTTP-specific, making it easier to adapt for other protocols beyond HTTP. This flexibility is particularly valuable for asynchronous communication scenarios, such as event-driven systems.

* Message Signatures, on the other hand, benefit from an existing HTTP-specific RFC with some established implementations. This existing groundwork means that this option could be simpler to deploy, to the extent such implementations are available and easily integrated.

* Given that the WIT (Workload Identity Token) is a type of JWT, the DPoP-inspired approach that also uses JWT is less complex and technology-intensive than Message Signatures. In contrast, Message Signatures introduce an additional layer of technology, potentially increasing the complexity of the overall system.

* Message Signatures offer superior integrity protection, particularly by mitigating message modification by middleboxes. See also Section 5.2.

* A key advantage of Message Signatures is that they support response signing. This opens up the possibility for future decisions about whether to make response signing mandatory, allowing for flexibility in the specification and/or in specific deployment scenarios.

* In general, Message Signatures provide greater flexibility compared to the DPoP-inspired approach. Future versions of this draft (and subsequent implementations) can decide whether specific aspects of message signing, such as coverage of particular fields,

should be mandatory or optional. Covering more fields will constrain the proof so it cannot be easily reused in another context, which is often a security improvement. The DPoP inspired approach could be designed to include extensibility to sign other fields, but this would make it closer to trying to reinvent Message Signatures.

Acknowledgments

The authors would like to thank Pieter Kasselman for his detailed comments, as well as Jason Costello, Maartje Eyskens and Radosナ B w Piliszek for implementing this draft and sharing their learnings.

We thank Daniel Feldman for his contributions to earlier versions of this document. We also thank Arndt Schwenkschuster and Brian Campbell who coauthored the grand unified WIMSE Workload to Workload protocol draft.

Authors' Addresses

Joe Salowey
CyberArk
Email: joe@salowey.net

Yaron Sheffer
Intuit
Email: yaronf.ietf@gmail.com