Use of Static-Static ECDH in JSON Object Signing and Encryption (JOSE)
draft-amringer-jose-ecdh-ss-00

Abstract

This document defines how to use the Static-Static mode of ECDH in
JSON Object Signing and Encryption (JOSE).

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 July 15, 2020.

Copyright Notice

Copyright (c) 2020 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 Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.
1. Introduction

The Internet Research Task Force (IRTF) JOSE Working Group defined the ECDH-ES as a key agreement mechanism in the JOSE context. This document defines how to use the ECDH key agreement mechanism in Static-Static mode in JOSE in an interoperable manner.

This document defines the conventions to use in the context of [RFC7516]

1.1. Notation and Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The JOSE key format ("JSON Web Key (JWK)") is defined by [RFC7517] and thumbprints for it ("JSON Web Key (JWK) Thumbprint") in [RFC7638].

2. Key Agreement with Elliptic Curve Diffie-Hellman Static-Static

This section defines the specifics of key agreement with Elliptic Curve Diffie-Hellman Static-Static, in combination with the Concat KDF, as defined in Section 5.8.2.1 of NIST.800-56A [1] for use as a symmetric key to wrap the CEK with the "C20PKW", "XC20PKW", "A128KW", "A192KW", "A256KW" algorithms, in the Key Agreement with Key Wrapping mode.
This mode is used as defined as the alternate way for ECDH-ES in Section 4.6.2 of RFC7518 [2] where the "apu" parameter MUST represent a random 512-bit value (analogous to PartyAInfo in Ephemeral-Static mode in [RFC2631]).

The following "alg" (algorithm) Header Parameter values are used to indicate that the JWE Encrypted Key is the result of encrypting the CEK using the corresponding algorithm:

<table>
<thead>
<tr>
<th>&quot;alg&quot; value</th>
<th>Key Management Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECDH-SS</td>
<td>ECDH-SS using Concat KDF for use directly as a symmetric key</td>
</tr>
<tr>
<td>ECDH-SS+C20PKW</td>
<td>ECDH-SS using Concat KDF and CEK wrapped with C20PKW</td>
</tr>
<tr>
<td>ECDH-SS+XC20PKW</td>
<td>ECDH-SS using Concat KDF and CEK wrapped with XC20PKW</td>
</tr>
<tr>
<td>ECDH-SS+A128KW</td>
<td>ECDH-SS using Concat KDF and CEK wrapped with A128KW</td>
</tr>
<tr>
<td>ECDH-SS+A192KW</td>
<td>ECDH-SS using Concat KDF and CEK wrapped with A192KW</td>
</tr>
<tr>
<td>ECDH-SS+A256KW</td>
<td>ECDH-SS using Concat KDF and CEK wrapped with A256KW</td>
</tr>
</tbody>
</table>

2.1. Header Parameters Used for ECDH Key Agreement

The following Header Parameters are used.

2.1.1. "spk" (Sender Public Key) Header Parameter

The "spk" (sender public key) value created by the originator for the use in key agreement algorithms. This key is represented either directly as a JSON Web Key [JWK] public key value, or encapsulated inside a JWE encoded using the compact serialization. The JWK MUST contain only public key parameters and SHOULD contain only the minimum JWK parameters necessary to represent the key; other JWK parameters included can be checked for consistency and honored, or they can be ignored. This Header Parameter MUST be present and MUST be understood and processed by implementations when an algorithm from this document is used.

2.1.2. "apu" (Agreement PartyUInfo) Header Parameter

The "apu" (agreement PartyUInfo) value for key agreement, represented as a base64url-encoded string. Its value contains a random 512-bit value. Use of this Header Parameter is REQUIRED. This Header
Parameter MUST be understood and processed by implementations when an algorithm from this document is used.

2.2. Header Parameters Used for Key Encryption

The following Header Parameters are used when the chosen "alg" algorithm includes a key encryption step.

2.2.1. "iv" (Initialization Vector) Header Parameter

The "iv" (initialization vector) Header Parameter value is the base64url-encoded representation of the 96-bit or 192-bit nonce value used for the key encryption operation. This Header Parameter MUST be present and MUST be understood and processed by implementations when an algorithm from this document is used.

2.2.2. "tag" (Authentication Tag) Header Parameter

The "tag" (authentication tag) Header Parameter value is the base64url-encoded representation of the 128-bit Authentication Tag value resulting from the key encryption operation. This Header Parameter MUST be present and MUST be understood and processed by implementations when these algorithms are used.

3. IANA Considerations

The following is added to the "JSON Web Signature and Encryption Algorithms" registry:

- Algorithm Name: "ECDH-SS" o Algorithm Description: ECDH-SS using Concat KDF o Algorithm Usage Location(s): "alg" o JOSE Implementation Requirements: Optional o Change Controller: IESG o Specification Document(s): Section 2 of [RFC-THIS] o Algorithm Analysis Documents(s): [RFC8439]

- Algorithm Name: "ECDH-SS+C20PKW" o Algorithm Description: ECDH-SS using Concat KDF and "C20PKW" o Algorithm Usage Location(s): "alg" o JOSE Implementation Requirements: Optional o Change Controller: IESG o Specification Document(s): Section 2 of [RFC-THIS] o Algorithm Analysis Documents(s): [I-D.irtf-cfrg-xchacha]

- Algorithm Name: "ECDH-SS+XC20PKW" o Algorithm Description: ECDH-SS using Concat KDF and "XC20PKW" o Algorithm Usage Location(s): "alg" o JOSE Implementation Requirements: Optional o Change Controller: IESG o Specification Document(s): Section 2 of [RFC-THIS] o Algorithm Analysis Documents(s): [I-D.irtf-cfrg-xchacha]
4. References

4.1. Normative References


4.2. Informative References

[I-D.irtf-cfrg-xchacha]
Arciszewski, S., "XChaCha: eXtended-nonce ChaCha and
AEAD_XChaCha20_Poly1305", draft-irtf-cfrg-xchacha-01 (work
in progress), July 2019.

4.3. URIs


Author’s Address

Guillaume Amringer
Canada

Email: g.amringer@gmail.com