JSON Key Exchange

JSON Web Service Binding v1.0

<series>draft-hallambaker-json-key-exchange

<status>informational

<stream>independent

<ipr>trust200902

<author>Phillip Hallam-Baker

<surname>Hallam-Baker

<initials>P. M.

<firstname>Phillip

<organization>Comodo Group Inc.

<email>[philliph@comodo.com](mailto:philliph@comodo.com)

<also>http://mathmesh.com/Documents/draft-hallambaker-json-key-exchange.html

The JSON Key Exchange

# Introduction

This document describes a lightweight key agreement mechanism using between 2 and four Diffe-Hellman or Elliptic Curve Diffie-Hallman keys. The mechanism may be used establish a shared session key with authentication of any or none of the initiator and the responder.

The approach described is similar to that adopted in the X3DH Key agreement <info="X3DH"/> used in Signal.

The objective of the key exchange is limited to establishing a shared secret between two mutually authenticated parties that cannot be derived by any other parties and cannot be reconstructed by either of the parties after the ephemeral contributions have been deleted.

The key exchange is intended for use as one component in a multi-layer security approach in which comprehensive security is provided through use of encryption and authentication at multiple layers in the protocol stack.

<figuresvg="../Images/SecurityStack.svg">Multi-layer security

Specifically, this key exchange is intended for use at the presentation layer (e.g. authenticate and encrypt HTTP message bodies) to establish keys for authentication and optional encryption of messages in Web Service transactions.

Data Layer

Encryption of cryptographic keys, protocol configuration profiles.

Presentation Layer

Authentication of parties to a Web Service transaction.

Transport Layer

Protect metadata against interception at single point on the message path (link or node)

Link Layer

Protect messages against traffic analysis by means of interception at multiple points on the message path

# Definitions

This section presents the related specifications and standard, the terms that are used as terms of art within the documents and the terms used as requirements language.

## Requirements Language

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].

## Defined Terms

No terms of art are defined.

## Related Specifications

JSON Key Exchange is used extensively in the Mathematical Mesh and related protocols <norm="draft-hallambaker-mesh-architecture"/>.

## Implementation Status

The implementation status of the reference code base is described in the companion document <info="draft-hallambaker-mesh-developer"/>.

# Key Exchange Protocol

## Parameters

The following parameters are defined

Algorithm

The key exchange protocol. Diffie Hellman in discrete log, EdECDH25519 or EdECDH448.<norm="RFC8032"/>

Key Derivation Function

The key derivation function. This is always HKDF <norm="RFC5869"/>

Key Wrap Function

The Key Wrap function. This is always RFC3394 <norm="RFC3394"/>

It should be noted that the algorithm described makes use of the Edwards form of the curve and not the Montgomery form described in <info="RFC7748"/>. While these curves are isomorphic, implementations of the Montgomery ladder do not lend themselves easily to the approach shown.

## Notation

The notation adopted in <info="X3DH"/> is applied with minor modifications.

X ||Y

The concatenation of the byte sequence X followed by the byte sequence Y.

KE (PK1, PK2)

The result of performing the key exchange (Diffie-Hellman or Elliptic Curve Diffie Hellman) with the public parameters of PK1 and the private parameters of PK2.

PK1+PK2

The public, private key pair formed by combining the public, private parameters of PK1 and PK2.

Expand (PRK, info, L)

The KDF expansion function that derives a key of length L bits from the Pre Random Key PRK using the string info as the salt.

Extract (IKM, salt)

The KDF extraction function that derives a Pre-Random Key from the Initial Keying Material IKM and the salt value salt. If the salt value is specified as 0, the default salt of a string of zero bits the same length as the Pre-Random Key to be extracted is used.

## Roles

Client

The party that initiates the key exchange

Service

The party that responds to the key exchange

## Keys

IKC

The identification key of the client. The encoding of this key MAY include one or more credentials such as a Mesh personal profile or an X.509 certificate binding the key to an identity.

IKS

The identification key of the service. The encoding of this key MAY include one or more credentials such as a Mesh personal profile or an X.509 certificate binding the key to an identity.

EKC

The ephemeral key of the client.

EKS

The ephemeral key of the service.

## Derived Keys

The key derivation function is used to derive separate keys for different purposes as shown below. The value L is the number of bits requires to key the algorithm specified. The salt value used to derive the PRK from the IKM is either the default salt value (all zeros) or the previous Rekey value as described below.

AK (PRK)

Authentication Key = Expand (PRK, "authentication", L)

EK (PRK)

Encryption Key = Expand (PRK, "encryption", L)

RA (PRK)

Rekey Key = Expand (PRK, "rekey", L)

W (PRK)

Witness value= Expand (PRK, "witness", L)

## Initial Keying Request

A key exchange request is either an initial key exchange request or a rekeying request. An initial key exchange request MAY be issued at any time but a rekeying request cannot be sent until at least one initial keying request has been completed.

Rekey messages are authenticated under a Rekey shared secret established in a previous session. This may be the immediately preceding session or any prior session whose rekey token has not expired.

### Initial Request Message

The client sends their identity and ephemeral key to the service. { IKC, EKC }

If the request message is an initial keying request, a credential associated with the identity key MAY be provided. The request MAY be authenticated by means of a digital signature.

### Initial Response Message

The service calculates the IKM value as follows:

IKM = KE (IKC + EKC, IKS + EKS)

PRK = Extract (IKM, 0)

The service returns the values { IKS, EKS, W(IKM) } in a message authenticated under AK(IKM)

## Rekeying.

Unless the key agreement is performed to a device of restricted capability, rekeying imposes minimal load on client or server and can be performed often, particularly if the Ed25519 curve is used for rekeying.

The state required for rekeying is separate from the keys used to encrypt and/or authenticate messages. This allows an application to store the rekeying key between communication sessions without risk of compromising the confidentiality or integrity of messages.

The use of the chaining salt ensures that rekeying cannot compromise the security of an already established key, even if a weaker key exchange algorithm is used. Thus a client MAY use an Ed488 key to perform an initial key exchange and then preform rekey operations using a

### Rekeying Request Message

The client sends a new ephemeral key to the service. { EKC }

The request message MUST be authenticated under the rekeying key of the shared secret of any unexpired session previously agreed. This allows the credentials of the parties to be omitted while providing the advantage of establishing a fresh forward secrecy session.

### Rekeying Response Message

The service calculates the IKM value as follows:

IKM = KE (EKC, EKS)

The service returns the values { EKS, W(IKM) } in a message authenticated under AK(IKM)

PRK = Extract (IKM, RA')

Where RA' is the previous rekeying key output.

<include="../Examples/KeyExchangeExamples.md"/>

<include="../Generated/KeyExchange.md"/>

# Security Considerations

# IANA Considerations

The following registrations are required:

HTTP Content Coding Registry

jose-jwb

Well-Known URIs

/.well-known/srv/

[Or change registry to FCFS]