Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 5, 2017

T. Hardjono
MIT
N. Smith
Intel Corp

July 4, 2016

# Fluffy: Simplified Key Exchange for Constrained Environments
# draft-hardjono-ace-fluffy-03

Abstract

This document proposes a simplified key exchange protocol for the establishment of a symmetric key shared between two devices or entities within a constrained environment. The pair-wise key establishment is performed using the mediation of a trusted Simple Key Distribution Center (SKDC) entity. The protocol also supports the mediated distribution of a group-key among multiple devices or entities for the purposes of protecting multicast messages.

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 http://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 January 5, 2017.

Table of Contents

## 1. Introduction

This document proposes a simplified key exchange protocol for constrained environments for the establishment of a symmetric key shared between two constrained devices. The pair-wise key establishment is performed using the mediation of a trusted Simple Key Distribution Center (SKDC) entity. The protocol also supports the mediated distribution of a group-key among multiple devices or entities for the purposes of protecting multicast messages.

The simplified key exchange protocol is referred to here as "Fluffy" and is based on a reduced set of Kerberos [RFC4120] messages, adjusting the message flows, types and features to the needs and capabilities of constrained devices and environments. It does not seek to be backward compatible with Kerberos implementations.

The protocol aims to be independent of the underlying transport protocol, and as such the protocol messages are integrity-protected against modifications in-transit. Similar to Kerberos [RFC4120], messages that carry sensitive information (such as keys and/or keying material) are protected using authenticated-encryption. Non-sensitive fields of the messages are integrity-protected using checksums or keyed-hash in the manner of RFC3961. A separate specification will be developed to address in more detail these cryptographic aspects of the current proposed protocol.

Two families of protocol messages are defined here:

- Pairwise key establishment between two entities: When a client seeks to establish a pairwise shared key (called the session encryption key) with a service principal (SP), it invokes the mediation of the SKDC. A four (4) message flow among the client, SKDC and SP are used to establish the pairwise shared key. A further two messages are used to delete the key prior to its expiration.

- Group-shared key establishment among multiple entities: When a client (e.g. client#1) seeks to create a group-shared key (called the group encryption key), it invokes the SKDC to create the group-key, to retain a copy at the SKDC and to return a copy to the requesting client. The distribution of the group-key to other members of a multicast group uses a simple fetch/deliver model in

which new group members (e.g. client#2) must ask for a copy of the group-key from the SKDC.

An additional set of exchanges are introduced to support the delivery of a public key pair to a client entity, with or without an accompanying digital certificate.

The current simplified key exchange protocol does not address the initial secret establishment between an entity and the SKDC. This is referred to in RFC4120 and RFC6113 as "pre-authentication". We anticipate that many types of constrained devices would need to undergo "on-boarding" into an operational state within a constrained environment, and that the on-boarding process may include (directly or as a side-effect) the establishment of the initial secret between the new device and the SKDC already operating in the environment. Thus, for example, the on-boarding process of a device (e.g. door-lock) into a constrained environment (e.g. home basement) with an SKDC entity (e.g. within the alarm panel) may consist of the device and the SKDC running a Diffie-Hellman exchange with the assistance of the human owner. The topic of on-boarding and off-boarding of devices is outside the scope of the current specification.

In this specification we assume that a transport such as CoAP [RFC7252] will be deployed in constrained environments where the IP protocol is operating at the network layer. Environments that are using non-IP transport are out of scope currently for this specification.

The current protocol uses JSON [RFC7159] and CBOR [RFC7049] for its message format. This is in-line with the RESTful paradigm and provides the greatest flexibility for the current protocol to be integrated with other protocols such as OAuth2.0 [RFC6749] for authorization and UMA [UMACORE] for user-centric consent management.

Since the intended deployment environment for the current protocol is a constrained environment, devices and entities there are assumed to use the UUID as the basis for identification. How a device is assigned a UUID is out of scope for the current specification.

The current specification acknowledges that in certain types of constrained environments there is the need for devices to not only operate autonomously for long periods of time, but also for devices to have the capability to take-on different roles with respect to other devices in the environment. Thus, a device (D1) acting as a client to another device (D2) that is acting as an SKDC could also be acting as an SKDC for yet a third device (D3). Thus, the device D1 may have the capability to be both a client and SKDC depending on the operational environment.

As in many deployment environments generally, often security is a trade-off among several factors (e.g. usability, assurance levels, cost, economic risk/benefits, and others). As such, it is realistic to acknowledge that the degree of trustworthiness of an SKDC is dependent on the value of the data and connections within the deployment environment. Thus, an SKDC within a home environment may not be expected to feature the same level of resistance to attacks as an enterprise deployment of a Kerberos KDC.

## 1.1. Notational 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].

Unless otherwise noted, all protocol properties and values are case sensitive. JSON [JSON] data structures defined by this specification MAY contain extension properties that are not defined in this specification. Any entity receiving or retrieving a JSON data structure SHOULD ignore extension properties it is unable to understand. Extension names that are unprotected from collisions are outside the scope of this specification.

## 1.2. Terminology

The current specification seeks to share terminology as much as possible with the terminology defined for CoAP [RFC7252]. However, since the intended Application(s) play a crucial role within constrained networks, we also refer to terminology used by OAuth 2.0 and UMA. Note that within a given constrained network, an device make take multiple roles (client or server) depending on the exchange and layers of the exchange in which they participate.

client
: The client is the entity in a constrained environment seeking to share a key pair-wise with the service principal.

service principal
: The entity with whom the client seeks to establish a pair-wise symmetric key is refereed to as the service principal (SP). This terminology is used to avoid confusion as much as possible with the generic term "server" or "service".

simple key distribution center
: The simple key distribution center (SKDC) is the entity which mediates the establishment of the pair-wise shared key between the client and service principal.

miniticket
: This is the data structure that carries the symmetric key to be shared between the client and service principal.

receipt
: This is the data structure that carries the symmetric key from the SKDC to an entity (client or service principal).

authenticator
: This is the data structure that carries proof-of-possession of a shared symmetric key between two entities.

pair-wise shared key
: A pair-wise shared key (PSK) is symmetric key shared only between a client and service principal.

group shared key
: A group shared key (GSK) is symmetric key shared by two or more entities.

session encryption key
: The session encryption key is the symmetric key generated by the SKDC to be shared pair-wise between the client and the service principal. A session encryption key is an instance of a PSK.

group encryption key
: The group encryption key is the symmetric key generated by the SKDC to be shared among members of a multicast group. A group encryption key is an instance of a GSK.

secret key
: The secret key is the symmetric key that is uniquely shared pair-wise between a client (or service principal) and the SKDC. This term is borrowed from RFC4120. Thus, the client secret key is the symmetric key that is uniquely shared pair-wise between the client and the SKDC. The SP secret key is the symmetric key that is uniquely shared pair-wise between the SP and the SKDC.

set-up keying material
: The cryptographic keying material (including possibly keys) resulting from the initial on-boarding process of a device into a constrained environment is referred to generally as "set-up keying material".

permissions and access control

The permissions and access control (PAC) is the set of information pertaining to permissions for entities within a constrained environment.

resource
The resource refers to the end-point at the service principal to which the application seeks access.

## 1.3. Design Considerations and Assumptions

There are a number of design considerations and background for the current protocol.

Transport: We assume that the entities in the constrained environment are deploying the CoAP protocol as transport [RFC7252]. However, the design of the current protocol seeks to be transport-independent as much as possible because we anticipate that not all constrained networks may be running CoAP.

JSON data structures: The data structures in this specification are expressed in JSON. We believe this provides the greatest flexibility for the protocol to be integrated into existing protocols for authorization (such as OAuth2.0 [Oauth2] and OpenID-Connect [OIDC]) and consent management by the resource/device owner (such as the User Managed Access (UMA) protocol [UMACORE]).

On-boarding and off-boarding: We assume that constrained devices will undergo the phase of "on-boarding" into a constrained environment. Similarly, "off-boarding" will be required when a constrained device leaves (or is removed) from a constrained environment. The notion of on-boarding is closely related to that of "take-ownership" of certain types of devices. Technologies such as the TPM [TPM] and EPID [EPID] play a crucial role in providing both cryptographic proof (technical trust) and human proof (social trust) in the process of changing the ownership of a device when it is activated and introduced into a constrained environment. We see a close relationship between on-boarding and the current protocol for establishing PSKs and GSKs within a constrained environment.

Secret key establishment or derivation: Following the on-boarding process of a client (resulting in the client and SKDC possessing set-up keying material), the client and the SKDC are assumed to generated the secret key which is shared pair-wise between the client and the SKDC. Methods include using PRFs and other one-way functions. The exact process of generating a secret key from the set-up keying material is out of scope of the current specification. As such, the current Fluffy protocol begins with

the assumption that each entity (client and service principal) already shares pair-wise a secret key with the SDKC. This secret key should be used only for key-management related messages as defined in this specification. Additionally, in this specification we have avoided the use of the term "long-term key" to refer to this secret key due to the broad meaning of this term.

Realms and zones: We have borrowed the notion of "realms" from RFC4120 because we anticipate that a constrained environment may consists of one or more physical networks, and which may be arranged (logically or physically) into "zones". Furthermore, we anticipate that in some use-cases the notion of a "realm" or "zone" may be more ephemeral than is commonly understood for RFC4120 deployments. Thus, there may be constrained use-cases where realms or zones are short-lived.

## 1.4. Out of Scope and Non-Goals:

The following are out of scope (non-goals) for the current specification:

Authorization and permissions: The issue of permissions and authorization is out of scope for this specification. However, the current specification anticipates the close integration between permissions, authentication and key establishment.

Discovery: Discovery of endpoints, identities, services and other aspects of a constrained environment is out of scope for the current specification.

Backward compatibility with Kerberos: It is not a goal of this specification to achieve backward compatibility with RFC1510 or RFC4120. Similarly, it is not the goal of this specification to be compatible with the MS-PAC [MSPAC] and MS-KILE [MSKILE] specifications.

Pre-authentication: RFC4120, RFC4556 and RFC613 uses the term "pre-authentication" to denote a client obtaining keying material for its secret key prior executing the Kerberos. It is not a goal of this specification to address pre-authentication.

Channel Binding for TLS and DTLS: Channel binding [RFC5929] for DTLS or TLS are out of scope in the current specification.

Certificate Issuance and Management: Issuance of X509 digital certificates and certificate management (in the sense of RFC2459, RFC2797 and RFC4210) is out of scope in the current specification.

## 2. Common Building Blocks

The current protocol employs a number of data structures that are common across several message types. A number of these data structures have semantic equivalents in RFC4120, while some are newly introduced.

Depending on the message type, some fields may be overloaded in its usage. For example, in the PSK-Request message the client states the identity and realm of the service principal within the SKDC-REQ-BODY. This is similar to RFC4120 because three (3) parties are involved in a PSK establishment (initiated by the client sending a PSK-Request message to the SKDC). However, when the SKDC-REQ-BODY is used in GSK establishment (initiated by an entity sending the SKDC a GSK-Request (GSK-REQ) message) the identity and realm fields are used instead to communicate the desired identity and the realm of the multicast group.

Two building blocks that do not have equivalents in RFC4120 are the Key Data and the Key Envelope structures:

Key Data: The keydata structure is used to convey cryptographic key(s) together with the associated operational parameters for the key(s). The structure of the keydata follows the JSON Web Key (JWK) definition of keys and keying material [RFC7517]. This is a departure from RFC4120.

Key Envelope: The key envelope is used to convey parameters related to a key (e.g. KeyID) but not the key itself. It is used in cases where entities need to refer to a key (e.g. group-key) without having to carry the key in the message.

In the following the common building blocks are discussed.

### 2.1. SKDC Request Body

The SKDC request body (SKDC-REQ-BODY) carries specific information regarding the type of request and the entities involved in the message. This data structure is used in the initial request send from a client (or SP) to the SKDC.

The SKDC-REQ-BODY varies slightly when used in the PSK-REQ and GSK-REQ messages.

SKDC-REQ-BODY:

- Key Type (kty): This denotes the type of key being requested by the sender (client or SP) of the request.

- Desired algorithm (etype): This is the algorithm that is desired (or supported) by the sender of the request (client or SP) .

- SKDC options - optional (skdc-options): These are flags that are intended for the SKDC only. This field is optional.

- SKDCs realm (skdcrealm): This the name of the realm, domain or zone of the SKDC.

- SKDC's identity (skdcname): This is the identity of the SKDC.

- Service principal's realm - optional (sprealm): This the name of the realm, domain or zone of the service principal in a PSK-REQ message. In a GSK-REQ message it is the realm of the multicast group. This field is optional.

- Service principal's identity (spname): In a PSK-REQ message this is the identity of the service principal. In a GSK-REQ message it is the identity or name of the multicast group.

- Client permissions - optional (cpac): In a PSK-REQ message this is the permisions desired by the client for itself. In a GSK-REQ message this is the permisions desired by the client for the multicast group. This field is optional.

### 2.2. Miniticket

The miniticket is always created by the SKDC and is always intended for the service principal (although it is delivered via the client who initiates with the PSK-REQ message). The SKDC responds to the client by sending a PSK-Response (PSK-REP) message containing the miniticket and a receipt. The miniticket contains a copy of the session encryption key to be delivered to the service principal by the client in a PSK-Establish (PSK-ESTB) message. As such, the sensitive parts (enc-part) of the miniticket is encrypted using the service principal's secret key (which it shares pair-wise with the KDC). The miniticket is functionally equivalent to the service-ticket in RFC4120.

The miniticket contains the following:

- Issuing SKDC's realm (skdcrealm): This the name of the realm, domain or zone of the SKDC that issued this miniticket.

- Issuing SKDC's identity (skdcname): This is the identity of the SKDC that issued this miniticket.

- Service principal's realm - optional (sprealm): This the name of the realm, domain or zone of the service principal for whom this miniticket is destined. This field is optional.

- Service Principal's identity (spname): This is the identity of the service principal for whom this miniticket is destined.

- Encrypted miniticket part (enc-part): This is the encrypted part of the miniticket intended for the service principal. It is encrypted using the secret key shared pair-wise between the SKDC and the service principal. The encrypted part contains the following:

  * Ticket flags - optional (tflags): This is the flags set by the SKDC for the service principal concerning this ticket. This field is optional.

  * Client's realm (crealm): This the name of the realm, domain or zone of the client with whom the receiver of this miniticket shares the enclosed key.

  * Client's identity (cname): This is the identity of the client with whom the receiver of this miniticket shares the enclosed key.

  * Client permissions - optional (cpac): This is the permissions and access control (PAC) structure containing permissions granted to the client associated with the enclosed key. This field is optional.

  * Time of authentication (authtime): This is the time at the SKDC when it created this miniticket in response to a request.

  * Expiration time of key (endtime): The is the expiration time of the key in this miniticket.

  * Service principal permissions - optional (sppac): This is the permissions and access control (PAC) structure granted to the service principal associated with the enclosed key. This field is optional.

  * Transited realms - optional (transited): This is the set of SKDCs and realms that was involved in the issuance of the miniticket for the service principal. This field is used for cross-realm ticket issuance. This field is optional.

  * Key data (keydata): This fields contains the key-data structure that carries the cryptographic key and other parameters

necessary for operating the key. See section Section 2.6 for the key-data structure.

## 2.3. Receipt

The receipt is always created by the SKDC and is used for the SKDC to deliver a key to a requesting party (be it client, service principal or multicast group members). The receipt is functionally equivalent to the SKDC-Response part in RFC4120.

In The PSK establishment flows the receipt is used to deliver the new session encryption key to the requesting client in a PSK-REP message from the SKDC.

In The GSK establishment flows the receipt is used to deliver the new group encryption key to the requesting entity using the GSK-Response (GSK-REP) message. Similarly, members of a multicast group must individually request a copy of the group key using the GSK-Fetch message (GSK-FET), to which the SDKC will send a receipt structure containing a copy of the group key via the GSK-Deliver (GSK-DLVR) message.

When used in a PSK-REP message as a response to the client's request for a new session encryption key, the receipt names the service principal who is to share the key with the client. The service principal identified in this receipt is the same as that stated in the matching miniticket.

When used in a GSK-REP message for a new group-key creation, the receipt instead names the multicast group with associated with this key. Note that the GSK-REP message has no accompanying miniticket because the SKDC is reponding solely to the requester of the new group-key in a 2-party flow. The receipt in a GSK-Deliver (GSK-DLVR) message (to deliver a copy of the group-key to members of the multicast group) is functionally identical to the receipt in the GSK-REP message.

A note about convention: in the remainder of this specification the entity that requests the creation of a group-key is denoted as the service principal (SP). The members of the multicast group are denoted as the client.

Receipt:

- Receipts flags - optional (rflags): This is the flags set by the SKDC concerning this receipt. This field is optional.

- Issuing SKDC's realm (skdcrealm): This the name of the realm, domain or zone of the SKDC that issued this receipt.

- Issuing SKDC's identity (skdcname): This is the identity of the SKDC that issued this receipt.

- Realm - optional (sprealm or mcastrealm): This field is optional, and is used as follows:

  * sprealm: In a PSK-REP message, sprealm is used. This is the name of the realm, domain or zone of the service principal with whom the client (recipient of this receipt) shares the enclosed session encryption key.

  * mcastrealm: In a GSK-REP message and GSK-DLVR message, the mcastrealm is used. This is the realm of the multicast group associated with the enclose group encryption key.

- Name of entity sharing this key (spname or mcastname):

  * spname: In a PSK-REP message, spname is used. This is the identity of the service principal who will be sharing the enclosed session encryption key with requesting client.

  * mcastname: In a GSK-REP message and GSK-DLVR message, mcastname is used. This is the identity of the multicast group associated with the enclosed group encryption key.

- Service Principal's SKDC - optional (spskdc): This field is optional. In a PSK-REP message, this is the identity of the service principal's SKDC. This field is absent in receipts used in a GSK-REP message or GSK-DLVR message.

- Permissions - optional (cpac or grppac): This field is optional. This is the permissions and access control (PAC) structure containing permissions granted, associated with the enclosed key.

  * cpac: In a PSK-REP message, the permissions pertains to the client who requested the session encryption key.

  * grppac: In a GSK-REP message or GSK-DLVR message, the permissions pertain to the members of the multicast group who share this common group encryption key.

- Time of authentication (authtime): This is the time at the SKDC when it created this receipt.

- Nonce from the sender's request (nonce): This is the nonce found in the previous request message (either PSK-REQ message or GSK-REQ message).

- Expiration time of key (endtime): The is the expiration time of the key in this receipt.

- Key data (keydata): This fields contains the key-data structure that carries the cryptographic key and other parameters necessary for operating the key. In a PSK-REP message, this key is the session encryption key to be shared between the client and service principal. In a GSK-REP message or GSK-DLVR message this is the group encryption key to be shared by the multicast group members. See section Section 2.6 for the key-data structure.

## 2.4. Authenticator

The authenticator is used by a sender to provide proof-of-possession (POP) to a receiver of a given key that the sender shares with the receiver. The authenticator here is functionally equivalent to the authenticator in RFC4120.

In the PSK-REQ and GSK-REQ messages, the requesting entity uses the authenticator to "authenticate" itself to the SKDC by providing proof-of-possession of the secret key which it shares pair-wise with the SKDC. Note that this mode of usage of the authenticator departs from RFC4120 where a key request message (namely the AP-REQ in RFC4120) is not accompanied by an authenticator.

In the PSK establishment flows, the authenticator is used also in the PSK-ESTB message that is sent from the client to the service principal. More specifically, in the PSK-ESTB message the client encrypts the authenticator using the session encryption key that the client obtained in the previous PSK-REP message it received from the SKDC. Here the authenticator proves to the service principal that the client knows the session encryption key that is enclosed in the accompanying miniticket.

The authenticator is also used in the PSK deletion and GSK deletion flows where the SKDC accompanies its PSK-DELT and GSK-DELT messages respectively with an authenticator that proves to the intended recipient of the message that the SKDC knows the secret key shared pair-wise with that recipient. As such, the authenticator can be used as a generic mechanism to provide proof-of-possession of a shared key between two entities.

Using the example of a client sending an authenticator to a service principal, the authenticator contains the following:

- Client's realm (crealm): This the identity of the realm, domain or zone of the client that created the authenticator.

- Client's identity (cname): This is the identity of the client that created the authenticator.

- Client's current time (ctime): This is the time at the client when it created the authenticator.

- Nonce (nonce): This is a new nonce generated by the client (for the recipient of the authenticator).

- Sequence number - optional (seqnum): This is the sequence number used by the client to detect attacks. This field is optional.

- Checksum - optional (cksum): This is the keyed-checksum (based on the session encryption key) used by the client as sender. This field is optional.

## 2.5. Acknowledgement

The acknowledgement structure (ack) is used by one entity to send a positive acknowledgement to another entity regarding a message that was previously exchanged. The acknowledgement would carry a nonce that was found in the related previous message. The type of acknowledgement is expressed in the enveloping header message (e.g. PSK-ACK type acknowleges a previous PSK-ESTB message).

Note that the acknowledgement structure is intended to be generic, and as such can also be used by the SKDC to send an acknowledgement to a client or service principal.

Using the example of a service principal (sender) sending the an acknowledgement to a client (receiver), the acknowledgement structure contains the following:

- Service principal's realm - optional (sprealm): This the identity of the realm, domain or zone of the service principal who created this acknowledment. This field is optional.

- Service Principal's identity (spname): This is the identity of the service principal for who created this acknowledgement.

- Nonce from the client's previous message (nonce): This is the nonce found in the previous message being acknowledged.

- Time of authentication (authtime): This is the time at the service principal when it created this acknowledgement message.

- Sequence number - optional (seqnum): This is the sequence number used to detect attacks. This field is optional.

### 2.6. Key Data

The key-data structure is used to carry cryptographic keys and the related parameters needed to operate the keys. The construction of the key-data structure follows that of the JSON Web Keys [RFC7517] which supports not only symmetric keys, but also public key pairs and X509 certificates.

The intent is to support the delivery of either a single symmetric key, a public key pair or one key (half) of a public key pair.

Delivery of an array of keys is for future consideration.

#### 2.6.1. Symmetric Key Data

The key-data structure for carrying a symmetric key is as follows.

- Key type (kty): This is the type of key contained in the current key-data structure.

- Key ID - optional (keyid): This is the name or handle for the key that can be be referred to by parties sharing they key. This field is optional.

- Key (key): This is the symmetric key.

- Key operations parameters (keyops): This is parameters required to operate the cryptographic key.

- Algorithm (alg): This is the algorithm name for the use of the key.

#### 2.6.2. Asymmetric Key Data

The key-data structure for carrying an asymmetric key and parameters is as follows.

- Key type (kty): This is the type of key contained in the current key-data structure.

- Key ID - optional (keyid): This is the name or handle for the key that can be be referred to by parties sharing they key. This field is optional.

- Key (key): This is the public key.

- Key operations parameters (keyops): This is parameters required to operate the cryptographic key.

- Algorithm (alg): This is the algorithm name for the use of the key.

- Public key usage (pkuse): For public keys this field indicates the use of the key. See RFC7517.

- X509 URL parameter (x5u): This parameter is a URI [RFC3986] that refers to a resource for an X.509 public key certificate or certificate chain [RFC5280]. See RFC7517.

- X509 certificate chains parameter - optional (x5c): This parameter contains a chain of one or more PKIX certificates [RFC5280]. See RFC7517.

- X509 thumbprint parameter - optional (x5t): This parameter contains a base64url-encoded SHA-1 thumbprint (a.k.a. digest) of the DER encoding of an X.509 certificate [RFC5280]. See RFC7517.

- X509 SHA256 thumbprint parameter - optional (x5t256): This parameter contains a base64url-encoded SHA-256 thumbprint (a.k.a. digest) of the DER encoding of an X.509 certificate [RFC5280]. See RFC7517.

## 2.7. Key Envelope

The key envelope structure is used in messages that refer to a cryptographic key by its KeyID. As such, the key envelope structure by definition must never carry any cryptographic keys or keying material.

The key envelope is used primarily in the deletion of a a PSK or a GSK. When the SKDC wishes to delete a given PSK that it shares with an entity, it can refer to the target key by way of the KeyID in the key envelope.

In order to delete a PSK that a client and service principal shares (through the mediation of the SKDC via a previous 3-party PSK establishment flow), the SKDC must separately delete the PSK at the client and at the service principal (i.e. using two separate PSK-Delete (PSK-DELT) messages).

The key envelope is encrypted using the secret key shared between the SKDC and the entity (client or service principal) for whom the envelope is destined.

The key envelope must never be encrypted using the target key that is to be deleted.

The key envelope structure contains the following:

- Envelope Flags - optional (envflags): This is the flags related to the envelope. This field is optional.
- Issuing SKDC's realm (skdcrealm): This the name of the realm, domain or zone of the SKDC that issued this envelope.
- Issuing SKDC's identity (skdcname): This is the identity of the SKDC that issued this envelope.
- Current time (authtime): This is the time at the SKDC when it created the encrypted envelope.
- Nonce (nonce): This is a new nonce generated by the SKDC (for the recipient of the envelope).
- Sequence number - optional (seqnum): This is the sequence number to detect attacks. This field is optional.
- Key data (keydata): This is the key-data structure which contains the KeyID of the target key. The key-data in a key envelope must not contain any cryptographic keys. See section Section 2.6 for the key-data structure.

## 3. Pair-wise Shared Key Establishment

This section describes the pair-wise key establishment between the client and the service principal.

### 3.1. Basic Protocol Exchange

Prior to executing the Fluffy protocol, a client must first be in possession of a secret key that it shares pair-wise with the SKDC. The process or method of obtaining the client secret key is outside the scope of the current specification, but for a device operating within a constrained environment this may be a direct consequence of the on-boarding or take-ownership process.

The PSK establishment consists of a 4-message flow from the client to SKDC, the SKDC back to the client, and between the client and the service principal.

Note that unlike RFC4120, the client must provide an authenticator in its first message (PSK-Request) to the SKDC. This authenticator is

encrypted by the client using the secret key it shares pair-wise with the SKDC.

The message flows consists of the following steps and are summarized in Figure 1.

- PSK-Request (PSK-REQ): The client sends a PSK-Request message to the SKDC asking for the SKDC to mediate the sharing of a new session encryption key between the client and the service principal. The client must indicate the intended service principal in this message. Unlike RFC4120 the client must include an authenticator that is encrypted to the SKDC as a proof of possession of the secret key it shares with the SKDC.

- PSK-Response (PSK-REP): The SKDC responds by generating a new session encryption key and placing the key into a miniticket intended for the service principal. The miniticket is encrypted using the secret key which is pair-wise shared only between the SKDC and the service principal. Additionally, the SKDC places a copy of this new session encryption key into a receipt structure, and encrypting it using the secret key pair-wise shared between the SKDC and the client. Both the miniticket and the receipt are then returned to the client. The client must forward the miniticket unmodified to the service principal in the PSK-Establish message.

- PSK-Establish (PSK-ESTB): The client then decrypts the receipt to obtain the session encryption key. The client uses this session encryption key to encrypt an authenticator structure as proof of possession for the service principal. The client then sends the authenticator and the miniticket (unmodified from the SKDC) to the service principal. The service principal decrypts the miniticket to obtain the session encryption key, and then it uses the session encryption key to verify the authenticator (by decrypting it). At this point the client and the service principal shares the session encryption key.

- PSK-Acknowledge (PSK-ACK): The service principal exercises the newly received session encryption key by encrypting an acknowledgement message to the client.

Similar to RC4120, the integrity of the messages containing cleartext data is protected using a checksum mechanism (e.g. keyed hash) based on the client's secret key [RFC3961].

The PSK Establishment Flows

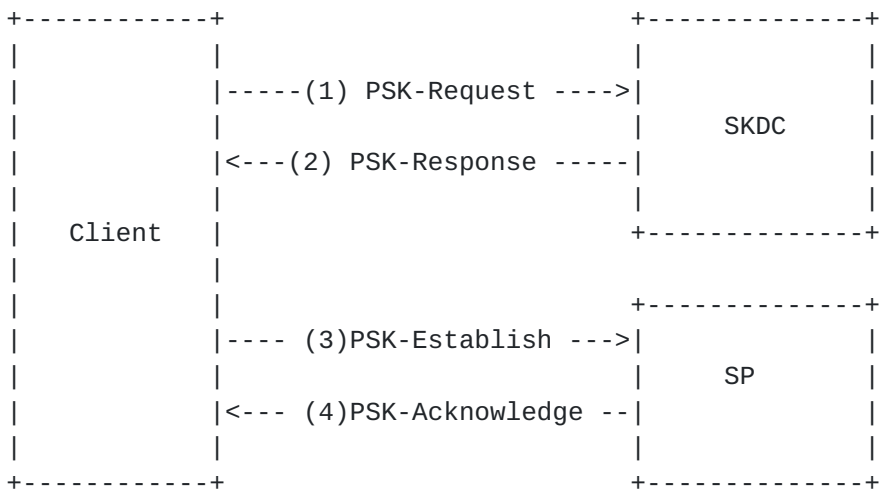


Figure 1

The message components as used in the protocol are summarized in Figure 2. Note that all protocol messages are integrity-protected, and some are encrypted.

The PSK Message Components

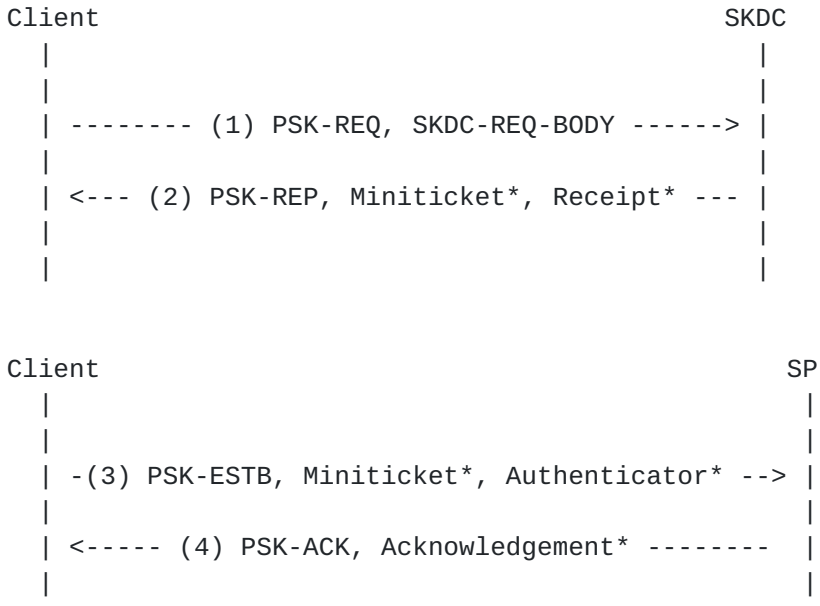


Figure 2

### 3.2. PSK-Request Message (PSK-REQ)

The PSK-Request (PSK-REQ) message is sent from the client to the SKDC asking for the SKDC to mediate the establishment of a pair-wise shared key between the client and the service principal. The client must indicate the intended service principal in this message.

- Protocol version (pvno): This the version of the protocol.
- Message type (msg-type): The message type for this message is "PSK-REQ".
- SKDC request body (req-body): The request body contains the parameters required by the SKDC to mediate key establishment for the client. See Section Section 2.1 for more information on the SKDC request body. The SKDC request body of a PSK-REQ message contains the following:
  - Key Type (kty): This is type of key being requested by the client from the SKDC.
  - Desired algorithm (etype): This is the algorithm that is desired (or supported) by the client.
  - SKDC options - optional (skdc-options).
  - SKDC's realm (skdcrealm).
  - SKDC's identity (skdcname).
  - Service principal's realm - optional (sprealm).
  - Service principal's identity (spname).
  - Client permissions - optional (cpac).
- Authenticator (authenticator): This is the authenticator encrypted by the client to the SKDC using the secret key that the client shares with the SKDC. See Section Section 2.4 for more information on the authenticator structure.
- Extensions - optional (ext): Reserved for future extensions. This field is optional.

### 3.3. PSK-Response Message (PSK-REP)

The PSK-Response (PSK-REP) is sent by the SKDC to the client as a response to the client's previous PSK-REQ message. The PSK-REP message carries two crucial data structures, namely the miniticket and the receipt.

The miniticket here is intended solely for the service principal and carries a copy of the session encryption key (key) intended for the service principal. As such the miniticket is encrypted to the service principal using the secret key shared between the SKDC and service principal. Although the miniticket is returned to the client, the client is unable to view or modify the encrypted parts of the miniticket.

The receipt here is intended solely for the client and carries a copy of the session encryption key (key) intended for the client. The receipt is encrypted to the client using the secret key shared between the SKDC and client.

The PSK-REP message contains the following:

- Protocol version (pvno): This the version of the protocol.
- Message type (msg-type): The message type for this message is "PSK-REP".
- Client's realm (crealm): This the name of the realm, domain or zone in which the client belongs in connection to this request.
- Client's identity (cname): This is the identity of the client.
- Miniticket (mticket): The miniticket contains the following:
  - Issuing SKDC's realm (skdcrealm).
  - Issuing SKDC's identity (skdcname).
  - Service principal's realm - optional (sprealm).
  - Service Principal's identity (spname).
  - Encrypted miniticket part (enc-part): This is the part of the PSK-REP message that is encrypted by the SKDC to the service principal, using the secret key that the SKDC shares pair-wise with the service principal. It contains the following:
    - Ticket flags - optional (tflags).

+ Client's realm (crealm): This the name of the realm, domain or zone in which the client belongs in connection to this request.

+ Client's identity (cname).

+ Client permissions - optional (cpac).

+ Time of authentication (authtime).

+ Expiration time of this key (endtime).

+ Service principal permissions - optional (sppac).

+ Transited realms - optional (transited).

+ Key data (keydata): This key-data structure contains a copy of the session encryption key destined for the service principal. See section Section 2.6 for the key-data structure.

o Receipt (receipt): This is the part of the PSK-REP message that is encrypted by the SKDC to the client, using the secret key that the SKDC shares pair-wise with the client. It contains the following:

* Receipts flags - optional (tflags).

* Issuing SKDC's realm (skdcrealm).

* Issuing SKDC's identity (skdcname).

* Service principal's realm - optional (sprealm).

* Service Principal's identity (spname).

* Service Principal's SKDC - optional (spskdc).

* Client permissions - optional (cpac).

* Time of authentication (authtime).

* Nonce from the Client's previous PSK-REQ request message (nonce).

* Expiration time of this key (endtime).

  * Key data (keydata): This key-data structure contains a copy of the session encryption key destined for the client. See section Section 2.6 for the key-data structure.

- Extensions - optional (ext): Reserved for future extensions. This field is optional.

### 3.4. PSK-Establish Message (PSK-ESTB)

The PSK-Establish (PSK-ESTB) message is sent from the client to the service principal requesting it to share a key (i.e. the session encryption key). The PSK-ESTB message contains two parts. The first is the miniticket obtained by the client from the SKDC in the previous PSK-Response (PSK-REP) message.

The second is the authenticator created by the client. The authenticator is encrypted using the session encryption key which the client obtained in the receipt from the previous PSK-REP from the SKDC).

The authenticator proves to the service principal that the client is in possession of the session encryption key.

This PSK-ESTB message is sent by the client to the service principal. It contains the following:

- Protocol version (pvno): This the version of the protocol.

- Message type (msg-type): The message type for this message is "PSK-ESTB".

- Client's realm (crealm): This the name of the realm, domain or zone of the client.

- Client's identity (cname): This is the identity of the client.

- Miniticket (mticket): This is the miniticket structure (unmodified) that the client received from the SKDC in the previous PSK-REP message. See Section Section 3.3 for the miniticket in the PSK-REP message.

- Authenticator: The authenticator in the PSK-ESTB contains the following:

  * Client's realm (crealm).

  * Client's identity (cname).

  * Client's current time (ctime).

  * A new nonce generated by the client for the service principal (nonce).

  * Sequence number - optional (seqnum).

  * Checksum - optional (cksum).

- Extensions - optional (ext): Reserved for future extensions. This field is optional.

### 3.5. PSK-Acknowledge Message (PSK-ACK)

The PSK-Acknowledge (PSK-ACK) message is sent from the service principal to the client in response to the previous PSK-ESTB message.

The message contains an acknowledgement part that is encrypted by the service principal using the session encryption key (which the service principal obtained in the miniticket in the previous PSK-ESTB message).

The PSK-ACK message contains the following:

- Protocol version (pvno): This the version of the protocol.

- Message type (msg-type): The message type for this message is "PSK-ACK".

- Client's realm (crealm).

- Client's identity (cname).

- Acknowledgement (ack): This is the acknowledgement structure that contains the following:

  * Service principal's identity (spname).

  * Service principal's realm - optional (sprealm).

  * Nonce from the Client's previous PSK-ESTB message (nonce).

  * Time of authentication (authtime).

  * Sequence number (inremented) from PSK-ESTB message - optional (seqnum).

- Extensions - optional (ext): Reserved for future extensions. This field is optional.

## 4. Pair-wise Shared Key Deletion

The current protocol supports the proactive deletion by the SKDC of a pairwise shared key (PSK) prior to the expiration of the key. The target PSK to be deleted must be a PSK that a client and service principal had established through the mediation of the SKDC using the PSK establishment flow.

Only the SKDC has the authority to send a key-deletion message (PSK-DELT) to an entity (client or service principal). A client or service principal MUST ignore key-deletion messages which did not come from the SKDC.

The SKDC authenticates itself to the client (service principal) by including a key envelope that is encrypted using the secret key shared between the SKDC and the client (service principal). The key envelope identifies the target key to be deleted by its key-ID.

If a PSK-DELT message issued by the SKDC is received at the client after the named key has in fact expired, the client must still respond with a PSK-Delete-Confirm (PSK-DELC) message. This confirms to the SKDC that the named key no longer exists at the client.

In order to delete a PSK that a client and service principal shares (through the mediation of the SKDC via a PSK establishment flow), the SKDC must delete the PSK at the client and at the service principal separately (using two separate PSK-DELT messages respectively).

The message components as used in the protocol are summarized in Figure 3.

The PSK Delete Message Components

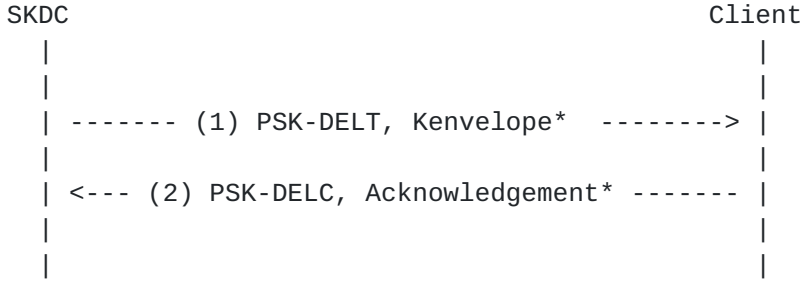


Figure 3

### 4.1. PSK-Delete Message (PSK-DELT)

The PSK-DELT message is sent from the SKDC to a client (or service principal) asking for the delition or erasure of the PSK identified in the message. The SKDC must indicate the intended recipient in this message.

- Protocol version (pvno): This is the version of the protocol.
- Message type (msg-type): The message type for this message is "PSK-DELT".
- Client's realm (crealm): This the identity of the realm, domain or group in which the client belongs in connection to this request.
- Client's identity (cname): This is the identity of the client.
- Key Envelope (kenvelope): The key envelope is encrypted using the client's secret key that it shared with the SKDC. The key envelope contains the following:
  - Envelope Flags - optional (envflags).
  - Issuing SKDC's realm (skdcrealm).
  - Issuing SKDC's identity (skdcname).
  - Current time (authtime).
  - New nonce generated by the SKDC (nonce).
  - Sequence number - optional (seqnum).
  - Key data (keydata): This key-data structure contains the KeyID of the target key to be deleted. The key-data in a key envelope must never contain a cryptographic key. See section Section 2.6 for the key-data structure.
- Extensions - optional (ext): Reserved for future extensions. This field is optional.

### 4.2. PSK-Delete-Confirm Message (PSK-DELC)

The psk-delete-confirm (PSK-DELC) message is sent from the client (or service) to the SKDC confirming the removal of the PSK identified in the previous PSK-DELT message. A client (or service principal) must only send the PSK-DELC message after it has successfully remove or erase the target key from its key store.

The acknowledgement in the PSK-DELC message is encrypted by the client using the secret key that the client shares with the SKDC. See Section Section 2.5 for more information on the acknowledgement structure.

The PSK-DELC message contains the following:

- Protocol version (pvno): This is the version of the protocol.
- Message type (msg-type): The message type for this message is "PSK-DELC".
- SKDC's realm - optional (skdcrealm).
- SKDC's identity (skdcname).
- Acknowledgement (ack):
  - Client's identity (cname).
  - Client's realm - optional (crealm).
  - Nonce from the SKDC's previous PSK-DELT message (nonce).
  - Client's current time (authtime).
  - Sequence number - optional (seqnum): This field is optional.
- Extensions - optional (ext): Reserved for future extensions. This field is optional.

## 5. Group Shared Key Establishment

The current protocol supports the establishment of a group-shared symmetric key (referred to as the "group encryption key" or "group-key") among a number of entities within a constrained environment. The group encryption key affords group-authenticity for messages but not source-authenticity since the symmetric key is shared among multiple entities (members of the multicast group). See RFC3740 for further discussion regarding multicast group security.

In the following we use the notation of the service principal (SP) as the entity that initiates the multicast group creation by requesting the SKDC to create a new group encryption key and to maintain a copy of that group-key until such time it expires or is deleted. The clients that request and obtain a copy of the group-key are denoted as "members" of the group. The current specification follows the

convention that only the group-creator and the SKDC are permitted to proactively delete a group encryption key.

Using the service principal as the group-creator, the service principal must accompany its request to the SKDC with an authenticator.

Each group encryption key is associated with an owner (creator) who requested its creation at the SKDC. When an SP seeks to establish a new group encryption key, it sends a GSK-Request message to the SKDC asking that the SKDC generate a new symmetric key (i.e. the group encryption key), return a copy of the group encryption key to the service principal (via a receipt inside a GSK-Response message) and for the SKDC to retain a copy of the group key (for subsequent fetches by the clients). The sensitive parameters of the GSK-Response message (including the group encryption key) inside the receipt is encrypted using the secret key pair-wise shared between the SP and the SKDC.

When a client seeks to obtain a copy of a group encryption key associated with a multicast group, the client sends a GSK-Fetch message to the SKDC identifying the multicast group (mcastname) of interest. The requesting client must accompany the request with an authenticator that is encrypted using the secret key shared between the client and the SKDC.

If a corresponding group or group encryption key does not exist at the SKDC, the SKDC returns an error message. Otherwise, the SKDC returns a copy of the group encryption key (inside a receipt) to the requesting client using the GSK-Deliver message. The sensitive parameters of the GSK-Deliver message (including the group encryption key) inside the receipt is encrypted using the secret key pair-wise shared between the requesting client and the SKDC.

The GSK Establishment Flows

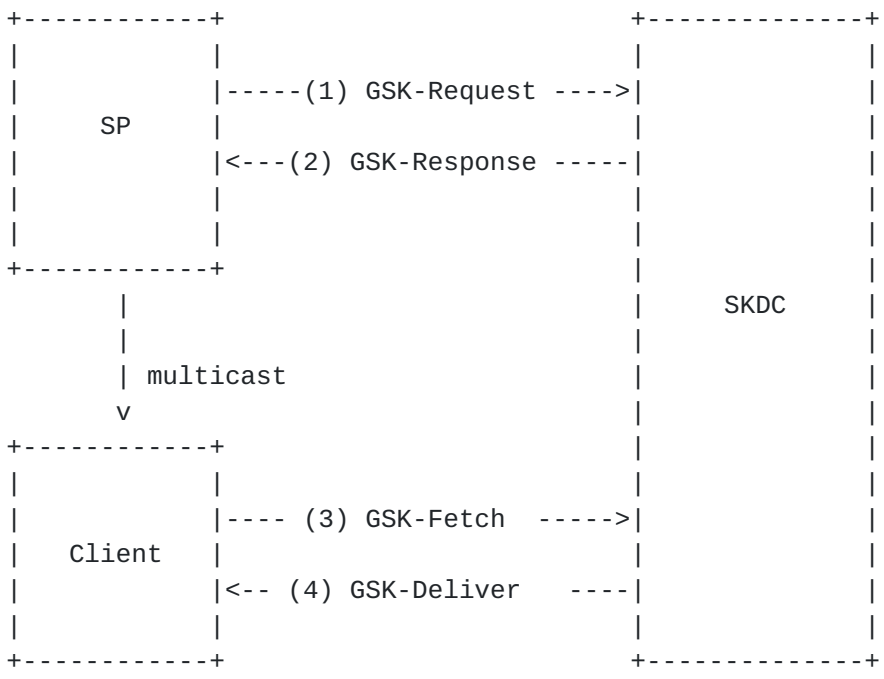


Figure 4

The GSK establishment in the protocol consists of two sets of 2-messages each:

- Creation of the group-key at the SKDC:
  - GSK-Request: A service principal sends a GSK-Request (GSK-REQ) message to the SKDC asking for a new group key to be created. The service principal provides the desired name (mcastname) of the multicast group. It must accompany the request with an authenticator to the SKDC. After generating the new group-key the SKDC retains a copy of the group-key (until it expires) and associates it with the multicast group name (mcastname).
  - GSK-Response: The requesting service principal obtains a copy of the new group-key via the GSK-Response (GSK-REP) message from the SKDC. A receipt structure to carries the group-key. The receipt is encrypted by the SKDC using the secret key it shares with the service principal.
- Fetching of a copy of the group-key from the SKDC:
  - GSK-Fetch: To request a copy of the group-key, a client sends the GSK-Fetch (GSK-FET) message to the SKDC with an

authenticator. The client must indicate the desired multicast group (mcastname) in the GSK-FET message.

* GSK-Deliver: The SKDC returns a copy of the group-key to the client via the GSK-Deliver (GSK-DLVR) message. A receipt structure to carries the group-key. The receipt is encrypted by the SKDC using the secret key it shares with the client.

The message components as used in the protocol are summarized in Figure 5. Note that all protocol messages are integrity-protected, and some are encrypted.

The GSK Message Components

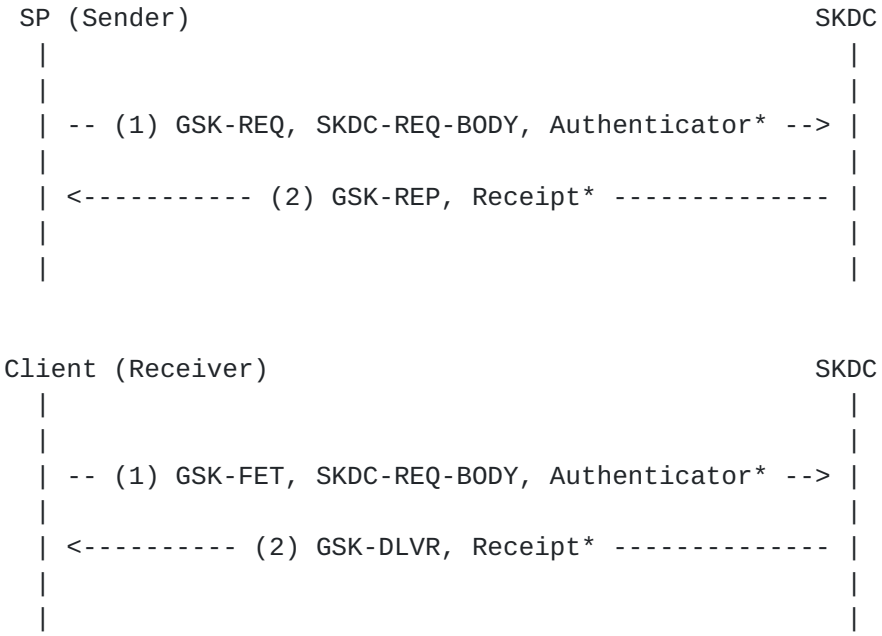


Figure 5

## 5.1. GSK-Request Message (GSK-REQ)

The GSK-Request message is sent from the service principal to the SKDC asking the SKDC to create a new group-key. The service principal authenticates itself to the SKDC by including an authenticator in the GSK-REQ message.

The contents of the GSK-REQ message is as follows:

o Protocol version (pvno): This the version of the protocol.

- Message type (msg-type): The message type for this message is "GSK-REQ".

- SKDC request body (req-body): The request body contains the parameters related to the group-key and multicast group. See Section Section 2.1 for more information on the SKDC request body. The SKDC request body in a GSK-REQ message contains the following:

  * Key Type (kty): This is type of key being requested. For a group shared key the type is "SYMM".

  * Desired algorithm (etype): This is the algorithm that is desired (or supported) by the service principal.

  * SKDC options - optional (skdc-options).

  * SKDC's realm (skdcrealm).

  * SKDC's identity (skdcname).

  * Multicast group realm - optional (mcastrealm): This is the desired realm name associated with the multicast group.

  * Multicast group identity (mcastname): This is the desired identity or name for the multicast group.

  * Group permissions - optional (grppac): This is the desired set of permissions associated with the multicast group.

- Authenticator: In the case of a GSK-REQ message, the authenticator is encrypted by the service principal (SP) using the secret key it shares with the SKDC. The authenticator in the GSK-REQ contains the following:

  * SP's realm (sprealm).

  * SP's identity (spname).

  * SP's current time (sptime).

  * A new nonce generated by the SP (nonce).

  * Sequence number - optional (seqnum).

  * Checksum - optional (cksum).

- Extensions - optional (ext): Reserved for future extensions. This field is optional.

### 5.2. GSK-Response Message (GSK-REP)

The GSK-Response (GSK-REP) message is sent from the SKDC to the service principal in response to the service principal's GSK-Request message. The GSK-Response message contains a receipt structure which carries the new group-key for the requesting service principal.

Note that the GSK-REP message does not contain a miniticket.

The Receipt in the GSK-Response message is encrypted by the SKDC to the requesting service principal using the secret key that is shared between the SKDC and the service principal.

The GSK-Response message contains the following:

- Protocol version (pvno): This the version of the protocol.
- Message type (msg-type): The message type for this message is "GSK-REP".
- SP's realm (sprealm): This the name of the realm, domain or zone in which the SP belongs in connection to this request.
- SP's identity (spname): This is the identity of the SP requesting the group-key in the previous GSK-REQ message.
- Receipt (receipt): The receipt carries the group-key and relevant parameters. It is encrypted by the SKDC to the SP using the secret key shared between the SKDC and the cSP. The receipt (receipt) contains the following:
  - Receipts flags - optional (rflags).
  - Issuing SKDC's realm (skdcrealm).
  - Issuing SKDC's identity (skdcname).
  - Multicast group realm - optional (mcastrealm).
  - Multicast group identity (mcastname).
  - Group permissions - optional (grppac).
  - Current time (authtime).
  - Nonce from the GSK-REQ request (nonce).
  - Expiration time of key (endtime).

  * Group key (keydata): The key-data structure contains the group-key destined for the requesting SP. See section Section 2.6 for the key-data structure.

- Extensions - optional (ext): Reserved for future extensions. This field is optional.

### 5.3. GSK-Fetch Message (GSK-FET)

The GSK-Fetch message is sent by a client to the SKDC asking for a copy of a group-key asociated with a multicast group. The client must identify the desired multicast group name (mcastname) in the SKDC-REQ-BODY of the message. The client authenticates itself to the SKDC by including an authenticator.

The contents of the GSK-Fetch message is as follows:

- Protocol version (pvno): This the version of the protocol.

- Message type (msg-type): The message type for this message is "GSK-FET".

- SKDC request body (req-body): The req-body contains the parameters required by the SKDC identify the multicast group. See Section Section 2.1 for more information on the SKDC request body. The SKDC request body of a GSK-FET message contains the following:

  * SKDC options - optional (skdc-options).

  * SKDC's realm - optional (skdcrealm).

  * SKDC's identity (skdcname).

  * Multicast group realm - optional (mcastrealm).

  * Multicast group identity (spname): This is the name of the multicast group whose group-key is being fetched.

  * KeyID - optional (keyid).

- Authenticator (authenticator): The authenticator in the GSK-FET is encrypted by the client to the SKDC using the secret key that the client shares with the SKDC. See Section Section 2.4 for more information the the authenticator structure. The authenticator in the GSK-FET contains the following:

  * Client's realm (crealm).

    * Client's identity (cname).

    * Client's current time (ctime).

    * A new nonce generated by the client (nonce).

    * Sequence number - optional (seqnum).

    * Checksum - optional (cksum).

- Extensions - optional (ext): Reserved for future extensions. This field is optional.

### 5.4. GSK-Deliver Message (GSK-DLVR)

The GSK-Deliver (GSK-DLVR) message is sent from the SKDC to the client in response to the client's GSK-Fetch message. The GSK-Deliver message uses the receipt structure to carry the group encryption key. The receipt is encrypted using secret key which is shared pair-wise between the client and the SKDC.

The contents of the GSK-Deliver message is as follows:

- Protocol version (pvno): This is the version of the protocol.

- Message type (msg-type): The message type for this message is "GSK-DLVR".

- Client's realm (crealm): This the name of the realm, domain or group in which the client belongs in connection to this request.

- Client's identity (cname): This is the identity of the client.

- Receipt (receipt): This is the part of the GSK-Deliver message carries the group-key. It is encrypted by the SKDC to the client using the secret key shared between the SKDC and the client. The receipt contains the following:

    * Receipts flags - optional (rflags).

    * Issuing SKDC's realm (skdcrealm).

    * Issuing SKDC's identity (skdcname).

    * Multicast group realm - optional (mcastrealm).

    * Multicast group identity (mcastname): This is the name of the multicast group whose group-key is being delivered.

  * Group permissions - optional (grppac).

  * Current time (authtime).

  * Nonce from the previous GSK-FET message (nonce).

  * Expiration time of key (endtime).

  * Group key (keydata): This key-data structure contains the group key destined for the requesting client. See section Section 2.6 for the key-data structure.

- Extensions - optional (ext): Reserved for future extensions. This field is optional.

## 6. Group Shared Key Deletion

The current protocol supports the proactive removal of a group-key associated with a multicast group prior to the expiration of the group-key. Only the creator (owner) of the multicast group can request the SKDC to delete a group-key (or the SKDC itself could perform a group-key deletion in response to an external authorized trigger).

Due to the nature of a symmetric group-key, the removal of a group-key from a multicast group requires the SKDC to issue unicast PSK-Delete messages to each known member of the group. When the SKDC sends the PSK-Delete message to a client who is a group member, the SKDC must identify the group-key via its KeyID. Each group member must respond to the SKDC with a PSK-Delete-Confirm (PSK-DELC) message to confirm the deletion or erasure of the group-key. See Section Section 4.1 for more information on the PSK-DELT and PSK-DELC messages.

The SKDC must wait for all known group members to individually confirm (via a PSK-DELC message) their deletion of the group-key. Only then should the SKDC return a GSK-Delete-Confirmation (GSK-DELC) message to the service principal who requested the group-key deletion.

When the SKDC is in the process of deleting a group-key, it must deny further requests for the group-key from new members.

The group-key deletion process involves four types of messages:

- GSK-Delete (GSK-DELT): The SP (as group-owner) sends a GSK-Delete request to the SKDC.

- PSK-Delete (KeyID): For each member of the group (i.e. clients who previosuly requested a copy of the group-key), the SKDC sends a unicast PSK-Delete (PSK-DELT) message to that client identifying the target key to be deleted (by KeyID). See Section Section 4.1 for the PSK-DELT message.

- PSK-Delete-Confirm (KeyID): Each group member responds after key-deletion with a PSK-DELC message to the SKDC. See Section Section 4.2for the PSK-DELC message.

- GSK-Delete-Confirmed (GSK-DELC): The SKDC responds with a GSK-Delete-Confirmed message to the SP (as group-owner) once all copies of the group-key has been deleted at the group-members.

The message flow for GSK deletion is shown in Figure 6.

The GSK Deletion Message Components

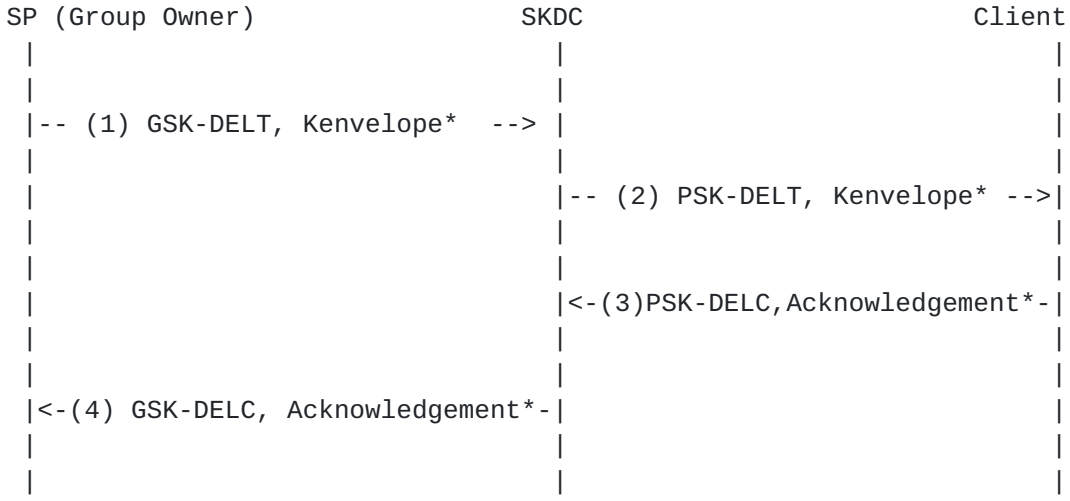


Figure 6

## 6.1. GSK-Delete Message (GSK-DELT)

The GSK-Delete (GSK-DELT) request message is sent from an SP (group owner) to the SKDC asking for the deletion or erasure of the group-key of the multicst group identified in the message.

The GSK-DELT message contains the following.

- Protocol version (pvno): This is the version of the protocol.

- Message type (msg-type): The message type for this message is "GSK-DELT".

- SP's realm (sprealm): This the name of the realm, domain or group in which the SP belongs in connection to this request.

- SP's identity (spname): This is the identity of the SP.

- Key Envelope (kenvelope): The key envelope is encrypted using the SP's secret key that it shares with the SKDC. The key envelope contains the following:

  * Envelope Flags - optional (envflags).

  * Multicast group realm - optional (mcastcrealm): This is the realm of the multicast group whose group-key is being deleted.

  * Multicast group identity (skdcname): This is the name of the multicast group whose group-key is being deleted.

  * Current time (authtime).

  * New nonce generated by the SP (nonce).

  * Sequence number - optional (seqnum).

  * Key data (keydata): This key-data structure contains the KeyID of the target key to be deleted. The key-data in a key envelope must never contain a cryptographic key. See section Section 2.6 for the key-data structure.

- Extensions - optional (ext): Reserved for future extensions. This field is optional.

### 6.2. GSK-Delete-Confirm Message (GSK-DELC)

In response to a GSK-Delete request from a service principal (as group owner), the SKDC sends a GSK-Delete-Confirm (GSK-DELC) message to the service principal for the successful deletion of not only its copy of the group-key, but also the successful deletion of the copies of the group-key at each group member (client).

The GSK-DELC message contains the following:

- Protocol version (pvno): This is the version of the protocol.

- Message type (msg-type): The message type for this message is "GSK-DELC".

- SP's realm - optional (sprealm).
- SP's identity (spname).
- Acknowledgement (ack): The acknowledgement is encrypted by the SKDC to the SP using the secret key shared only between the SKDC and SP.
  - Multicast group identity (mcastname): This is the name of the multicast group whose group-key has been deleted.
  - Multicast group realm - optional (mcastrealm).
  - Nonce from the SP's previous GSK-DELT message (nonce).
  - Current time (authtime).
  - Sequence number - optional (seqnum): This field is optional.
- Extensions - optional (ext): Reserved for future extensions. This field is optional.

## 7. Public Key Pair Establishment

The current protocol supports the distribution of public key pairs and certificates. Since the SKDC is not a certificate authority (in the sense of RFC2459), issuing (signing) X509 certificates is out of scope for the current specification. If the SKDC receives from a client a request for a new certificate, the SKDC must obtain a certificate from a CA server (or CA service) located either in the same realm/zone or elsewhere on behalf of the requesting client. How the SKDC discovers CA services is out of scope for the current specification.

In the following we provide additional message types to support a client requesting the SKDC to deliver a new public key pair and to return the key pair to the client.

When a client seeks to obtain a new public key pair, the client sends a Public Key Pair Request (PKP-REQ) message to the SKDC. The requesting client must include an authenticator that is encrypted using the secret key it shares with the SKDC. The SKDC returns the key-pair in the receipt structure to the client (encrypted to the client).

As a further option to the PKP-REQ message, if the client identifies a service principal in the SKDC-REQ-BODY of the PKP-REQ message, the SKDC would also return a miniticket that contains only the public

half of the key-pair. The miniticket would be encrypted by the SKDC to the named service principal, using the secret key that the SKDC shares with the service principal. Note that this is a departure from the PSK-Request semantics (for symmetric key requests) because the miniticket contains a public key (instead of a symmetric key).

The client (or the SKDC) then sends the miniticket to the service principal who can decrypt the miniticket using the secret key it shares with the SKDC.

By encrypting the miniticket to the service principal, the SKDC is effectively attesting to the binding between the public key pair and the client's identity (without the use of digital certificates). The service principal trusts SKDC to provide this pseudo-attestation regarding the client as the owner of the new public key pair.

Note that in this approach the security of the public key pair is only as a secure as the symmetric key algorithm used to encrypt the receipt and the miniticket.

If additionally the client seeks to obtain a digital certificate for the new public key, then the client must indicate this option in the SKDC-Options field (in the SKDC-REQ-BODY) of the PKP-Request message. There are several options available related to digital certificates and CA certificates (trust anchors):

No certificate: This is the default setting for the PKP-Request message.

Certificate: This means the client additionally requests the return of a new X509 certificate corresponding to the new public key.

Include certificate chain: This option is meaningful only of the client requests a new X509 certificate. When set, this option means that the client also requests the certificate chain consisting of copies of all signing certificates to the top of the certificate hierarchy.

### 7.1. Public Key Pair Request (PKP-REQ)

A client uses the PKP-Request message (PKP-REQ) to request a new public key pair from the SKDC. The client must include an authenticator when sending this message to the SKDC. The PKP-REQ message contains the following:

- Protocol version (pvno): This the version of the protocol.

- Message type (msg-type): The message type for this message is "PKP-REQ".

- SKDC request body (req-body): The request body contains the parameters required by the SKDC in the context of this request. the See Section Section 2.1 for more information on the SKDC request body. The SKDC request body of a PKP-REQ message contains the following:

  * Key Type (kty): This is type of key being requested by the client from the SKDC.

  * Desired algorithm (etype): This is the algorithm that is desired (or supported) by the client.

  * SKDC options - optional (skdc-options): The client sets the "certificate request" option (and possibly the "certificate chain" option) in this field if it requests a digital certificate (and corresponding chain) to be returned together with the public key pair.

  * SKDC's realm (skdcrealm).

  * SKDC's identity (skdcname).

  * Service principal's realm - optional (sprealm):

  * Service principal's identity -- optional (spname):

    + spname is present: If the spname is present, this means that the client wishes for the SKDC to prepare copy of the new public key only for the named service principal via the miniticket structure.

    + spname is absent: If the spname is absent the SKDC delivers the public key pair only to the client via the receipt structure.

  * Client permissions - optional (cpac).

- Authenticator (authenticator): This is the authenticator encrypted by the client to the SKDC using the secret key that the client shares with the SKDC. See Section Section 2.4 for more information on the authenticator structure.

- Extensions - optional (ext): Reserved for future extensions. This field is optional.

### 7.2. Public Key Pair Response (PKP-REP)

In response to the PKP-Request message (PKP-REQ) from a client entity, the SKDC returns a copy of the new public key pair to the client in a receipt structure, encrypted using the secret key shared between the client and SKDC. This ensures that only the client is in possession of the new public key pair (notably the private key).

If the SKDC-REQ-BODY of the client's previous PKP-request message contains the identity of a service principal (spname), the SKDC must also create a miniticket containing a copy of the public key only. The miniticket is encrypted to the service principal.

The PKP-REP message contains the following:

- Protocol version (pvno): This the version of the protocol.
- Message type (msg-type): The message type for this message is "PKP-REP".
- Client's realm (crealm): This the name of the realm, domain or zone in which the client belongs in connection to this request.
- Client's identity (cname): This is the identity of the client.
- Miniticket - optional (mticket): If the client identified a service principal in the previous PKP-REQ message (in its SKDC-REG-BODY), then a miniticket is included by the SKDC in this PKP-REP message. If present, the miniticket contains the following:
  - Issuing SKDC's realm (skdcrealm).
  - Issuing SKDC's identity (skdcname).
  - Service principal's realm - optional (sprealm).
  - Service Principal's identity (spname).
  - Encrypted miniticket part (enc-part): This is the part of the PKP-REP message that is encrypted by the SKDC to the service principal, using the secret key that the SKDC shares pair-wise with the service principal. It contains the following:
    - Ticket flags - optional (tflags).
    - Client's realm (crealm): This the name of the realm, domain or zone in which the client belongs in connection to this request.

+ Client's identity (cname).

+ Client permissions - optional (cpac).

+ Time of authentication (authtime).

+ Expiration time of this key - optional (endtime): This field is present if the SKDC returns a public key pair only. If a certificate accompanies the public key pair, then this field is absent.

+ Service principal permissions - optional (sppac).

+ Transited realms - optional (transited).

+ Key data (keydata): In a PKP-REP message, the key-data structure in the miniticket contains only the public key of the client. The private key must not be included. If the client had also requested a new certificate, the certificate is included here. See section Section 2.6 for the key-data structure.

o Receipt (receipt): This is the part of the PKP-REP message that is encrypted by the SKDC to the client, using the secret key that the SKDC shares pair-wise with the client. It contains the following:

* Receipts flags - optional (tflags).

* Issuing SKDC's realm (skdcrealm).

* Issuing SKDC's identity (skdcname).

* Service principal's realm - optional (sprealm).

* Service Principal's identity (spname).

* Service Principal's SKDC - optional (spskdc).

* Client permissions - optional (cpac).

* Time of authentication (authtime): This is the time of the creation of this receipt.

* Nonce from the Client's previous PSK-REQ request message (nonce).

* Expiration time of this key (endtime).

* Key data (keydata): In a PKP-REP message, the key-data structure in the receipt contains both the public key and private key belonging to the requesting client. If the client had also requested a new certificate, the certificate is included here. See section Section 2.6 for the key-data structure.

o Extensions - optional (ext): Reserved for future extensions. This field is optional.

## 8. JSON Message Format

TBD.

## 9. Encryption and Checksums

TBD.

## 10. Security Considerations

TBD.

## 11. Privacy Considerations

TBD.

## 12. IANA Considerations

TBD.

## 13. Acknowledgments

We thank Jesse Walker for design inputs and initial review.

## 14. References

### 14.1. Normative References

[JSON] Bray, T., "The JavaScript Object Notation (JSON) Data Interchange Format", March 2014, <https://tools.ietf.org/html/rfc7159>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC6347] Rescorla, E., "Datagram Transport Layer Security Version 1.2", January 2012, <http://tools.ietf.org/html/rfc6347>.

[RFC7252] Shelby, Z., "The Constrained Application Protocol (CoAP)", June 2014, <http://tools.ietf.org/html/rfc7252>.

### 14.2. Informative References

[ACE] Seitz, L., Ed., "ACE Use Cases", October 2012, <https://tools.ietf.org/wg/ace/draft-ietf-ace-usecases/>.

[BR-3KPD] Bellare, M. and P. Rogaway, "Entity Authentication and Key Distribution (In Advances in Cryptology, pages 110-125. Springer-Verlag, 1993)", September 1993, <http://link.springer.com/>.

[Choo04] Choo, K., Boyd, C., Hitchcock, Y., and G. Maitland, "On Session Identifiers in Provably Secure Protocols (Security in Communication Networks 4th International Conference, SCN 2004)", September 2004, <http://link.springer.com/>.

[Choo06] Choo, R., "Key Establishment: Proofs and Refutations", May 2006, <http://eprints.qut.edu.au/16262/1/Kim-Kwang_Choo_Thesis.pdf>.

[EPID] Brickell, E. and J. Li, "Enhanced Privacy ID (in NIST Privacy Enhancing Cryptography Conference 2011)", December 2011, <http://csrc.nist.gov/groups/ST/PEC2011/presentations2011/brickell.pdf>.

[MSKILE] Microsoft, ., "Kerberos Protocol Extensions (v20140502)", May 2014, <https://msdn.microsoft.com/en-us/library/cc233855.aspx>.

[MSPAC] Microsoft, ., "Privilege Attribute Certificate Data Structure (v20140502)", May 2014, <https://msdn.microsoft.com/en-us/library/cc237917.aspx>.

[NS] Needham, R. and M. Schroeder, "Using encryption for authentication in large networks of computers (CACM)", December 1978, <http://en.wikipedia.org/wiki/Needham-Schroeder_protocol>.

[OAuth2] Hardt, D., "The OAuth 2.0 Authorization Framework", October 2012, <http://tools.ietf.org/html/rfc6749>.

[OIDC] Sakimura, N., "OpenID Connect Core 1.0 incorporating errata set 1", November 2014, <http://openid.net/specs/openid-connect-core-1_0.html>.

[RFC3961] Raeburn, K., "Encryption and Checksum Specifications for Kerberos V5", February 2005, <http://tools.ietf.org/html/rfc3961>.

[RFC3986] Berners-Lee, T., "Uniform Resource Identifier (URI): Generic Syntax", January 2005, <http://www.ietf.org/rfc/rfc3986.txt>.

[RFC4120] Neuman, C., "The Kerberos Network Authentication Service (V5)", July 2005, <http://tools.ietf.org/html/rfc4120>.

[RFC6113] Hartman, S., "A Generalized Framework for Kerberos Pre-Authentication", April 2011, <http://tools.ietf.org/html/rfc6113>.

[TPM] TCG, ., "Trusted Platform Module (TPM) Main Specification Level 2 Version 1.2", March 2011, <http://www.trustedcomputinggroup.org/resources/tpm_main_specification>.

[UMACORE] Hardjono, T., Ed., "User-Managed Access (UMA) Profile of OAuth 2.0", November 2014, <https://docs.kantarainitiative.org/uma/draft-uma-core.html>.

## Appendix A. Document History

NOTE: To be removed by RFC editor before publication as an RFC.

Authors' Addresses

Thomas Hardjono
MIT

Email: hardjono@mit.edu

Ned Smith
Intel Corp

Email: ned.smith@intel.com