Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 24, 2015

T. Hardjono
MIT
N. Smith
Intel Corp

March 23, 2015

Fluffy: Simplified Key Exchange for Constrained Environments
draft-hardjono-ace-fluffy-00

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 September 24, 2015.

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.

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.

- 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.

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.

The security of current proposed protocol seeks to be independent of is underlying messaging transport, and its messages are protected independent of any underlying secure transport layer (such as TLS [RC5246] and DTLS [RFC6347]). However, 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.

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

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.

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 and TLS are out of scope in the current specification.

2. Pair-wise Shared Key (PSK) Establishment

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

2.1. 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 related to the on-boarding or take-ownership process.

For example, the manufacturer of the device may ship the device containing some fixed cryptographic parameter (e.g. Diffie-Hellman group and modulo values). When the new owner (e.g. home consumer) seeks to introduce the device into a constrained network, he or she may need to enter commands that result in the device becoming a client of second device already operating as the SKDC. We refer to the key resulting from on-boarding or take-ownership as the "set-up keying material". The set-up keying material in turn can be used to derive or generate the shared secret key at the client and the SKDC.

The Fluffy protocol 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. Authentication of the client to the SKDC is achieved by virtue of proof of possession by the client of the client's secret key. Authentication of the client to the service principal is achieved by proof of possession by the client of a copy of the session encryption key (which the SKDC issued for only the client and the service principal).

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

- PSK-Request: 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.

- PSK-Response: 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.

- PSK-Establish: 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 decrypt the authenticator. At this point the client and the service principal shares the session encryption key.

- PSK-Acknowledge: The service principal exercises the newly received session encryption key by encrypting a message for 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

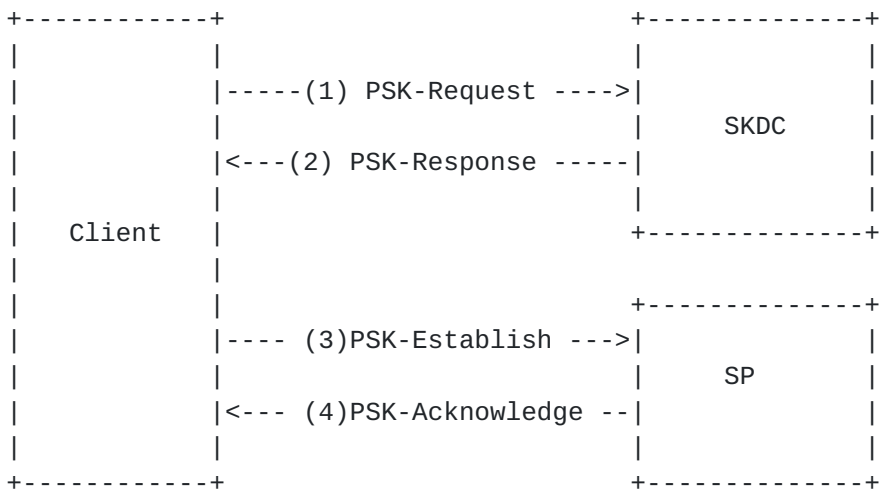

Figure 1

2.2. Common Building Blocks

The Fluffy protocol messages employ a number of data structures that are common across several messages. Many of these are borrowed from RFC4120 though not necessarily compatible with it. These are as follows:

- SKDC-request body: This is part of the request message that carries parameters regarding the client's identity, service principal's identity, the client's nonce and others.

- Receipt: The receipt is the data structure created by the SKDC to carry the key to be delivered to the client (either session-key or group-key). The receipt is always encrypted to the client by the SKDC using the client's secret key (shared pair-wise with the SKDC). The receipt is functionally equivalent to the SKDC-Response part in RFC4120. The contents of the receipt for PSK Establishment (session encryption key) is slightly different from the receipt use for GSK Establishment (group encryption key), with the later carrying more fields.

- Miniticket: The miniticket is the data structure created by the SKDC to carry the session encryption key to be delivered (via the client) to the service principal in the PSK Establishment. The miniticket has an encrypted-part which is encrypted to the service principal by the SKDC. The miniticket is functionally equivalent to the service-ticket in RFC4120.

- Authenticator: The authenticator is the data structure created and encrypted by the client with the aim of authenticating itself (by virtue of proof of possession) to the recipient of the authenticator. For the PSK Establishment the authenticator is encrypted by the client using the session encryption key shared between the client and the service principal. For the GSK Establishment the authenticator is encrypted by the client using the client's secret key which it shares with the SKDC. A time-stamp prevents replay attacks. The authenticator here is functionally equivalent to the authenticator in the AP-REQ message in RFC4120.

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

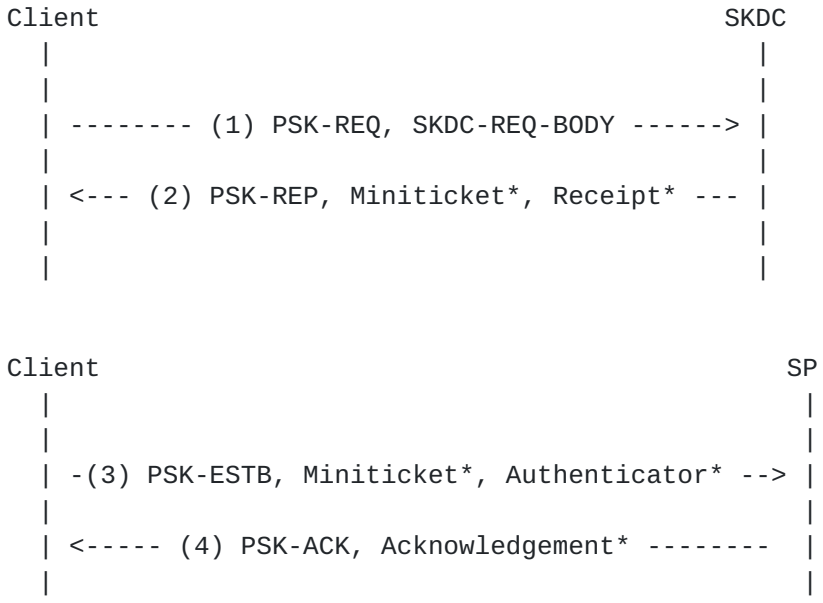

Figure 2

2.3. PSK-Request Message

The psk-request 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 2.3.1 for the SKDC request body.

2.3.1. SKDC request body

The SKDC request body contains the following:

- SKDC options - optional (skdc-options): These are optional flags that the client can set. This field is optional.

- 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.

- Service principal's identity (spname): This is the identity of the service principal.

- Service principal's realm - optional (sprealm): This the identity of the realm, domain or group in which the service principal belongs in connection to this request. This field is optional.

- Client's nonce (nonce): This is the nonce generated by the client.

- Desired encryption algorithm (etype): This is the encryption algorithm desired/supported by the client to be used with the service principal.

2.4. PSK-Response Message

This psk-response message is sent by the SKDC in response to a received psk-request message from a client. The psk-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 "PSK-REP".

- Client's realm (crealm): This the identity of the realm, domain or group in which the client belongs in connection to this request. This is obtained from the psk-request message.

- Client's identity (cname): This is the identity of the client obtained from the previous sdkc-request message.

- Miniticket (mticket): This is the miniticket structure that is intended for the service principal. See section Section 2.4.1 for the miniticket.

- Receipt (receipt): This is the part of the psk-response message that encrypted to the client. It is encrypted using the client's secret key that it shared pair-wise with the SKDC. See section Section 2.4.2 for the receipt.

2.4.1. Miniticket

The miniticket is always created by the SKDC and is always intended for the service principal. The miniticket contains a copy of the session encryption key to be delivered to the service principal. 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 contains the following:

- Issuing SKDC's realm (skdcrealm): This is the realm of the SKDC that issued the miniticket.

- Service Principal's identity (spname): This is the identity of the service principal.

- Service principal's realm - optional (sprealm): This the identity of the realm, domain or group in which the service principal belongs in connection to this request. This field is optional.

- 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:

  * Session encryption key (key): This is the symmetric key generated by the SKDC and to be shared pair-wise between the client and service principal.

  * 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.

* Time of authentication (authtime): This is the time at the SKDC when it received the psk-request message.

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

* Service principal permissions - optional (sppac): This is the permissions and access control (PAC) structure related to the service principal. This field is optional.

* Transited realms - optional (transited): This is the set of SKDC/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.

2.4.2. Receipt for PSK

The receipt is always created by the SKDC and is always intended for the client. The receipt contains a copy of the session encryption key to be delivered to the client. As such, the receipt is encrypted using the client's secret key (which it shares pair-wise with the KDC).

The receipt (receipt) contains the following:

- Issuing SKDC's realm (skdcrealm): This is the realm of the SKDC that issued the miniticket.

- Service principal's identity (spname): This is the identity of the service principal.

- Service principal's realm - optional (sprealm): This the identity of the realm, domain or group in which the service principal belongs in connection to this request. This field is optional.

- Session encryption key (key): This is the symmetric key generated by the SKDC and to be shared pair-wise between the client and service principal.

- Time of authentication (authtime): This value is the same as found in the encrypted miniticket part.

- Expiration time of key (endtime): This value is the same as found in the encrypted miniticket part.

- Nonce from the client's request (nonce): This is the nonce from the client's psk-request message.

- Client permissions - optional (cpac): This is the permissions and access control (PAC) structure related to the client. This field is optional.

2.5. PSK-Establish Message

The psk-establish message is sent from the client to the service principal requesting it to share a key (i.e. the session encryption key) with the service principal. The psk-establish message contains two parts. The first is the miniticket obtained from the SKDC in the previous psk-response 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 within the psk-response message). The authenticator authenticates the client to the service principal by virtue of the proof of possession (POP) of the session encryption key by the client to the service principal.

This psk-establish 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".

- Miniticket (mticket): This is the miniticket structure (unmodified) that the client received from the SKDC in the psk-response message. See Section 2.4.1 for the miniticket.

- Authenticator: In the case of a PSK-ESTB message, the authenticator is encrypted using the session encryption key obtained by the client in the receipt (within the psk-response message). See section Section 2.5.1 for the authenticator.

2.5.1. Authenticator

The authenticator is the data structure encrypted by the client with the aim of providing proof of possession (of either a PSK or GSK) to the recipient of the authenticator. It is used both in the PSK Establishment flows and the GSK Establishment flows.

For the PSK Establishment the authenticator is encrypted by the client using the session encryption key shared between the client and the service principal. The intended recipient of the authenticator in the PSK Establishment flows is the service principal.

For the GSK Establishment the authenticator is encrypted by the client using the client's secret key which it shares with the SKDC. The intended recipient of the authenticator in the GSK Establishment flows is the SKDC.

The authenticator contains the following:

- 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.
- Client's current time (ctime): This is the time of the client's clock.
- 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.

2.6. PSK-Acknowledge Message

The psk-acknowledge message is sent from the service principal to the client in response to the previous psk-establish message.

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

The psk-establish message is sent by the service principal to the client. 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-ACK".
- Acknowledgement (sp-ack): This is the acknowledgement structure that contains the following:
  - Service principal's identity (spname).
  - Client's current time (ctime).

* Sequence number - optional (seqnum): This is the incremented sequence-number, which was found in the previous psk-establish message. This field is optional.

3. Group Shared Key (GSK) Establishment

The current protocol supports the establishment of a group-shared key (referred to as the group encryption 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).

A client must posses a secret key (shared pair-wise with the SKDC) before the client can request the creation of a new group encryption key at the SKDC.

Each group encryption key is associated with an owner (creator) who requested its creation at the SKDC. When a client (e.g. Client#1) 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 client (via a receipt inside a GSK-Response message) and for the SKDC to retain a copy of the key (for subsequent fetches by other 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 client and the SKDC.

When a different client (e.g. Client#2) seeks to obtain a copy of a group encryption key belonging to another client (e.g. Client#1), the Client#2 sends a GSK-Fetch message to the SKDC identifying the multicast group (mcastname) of interest. 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

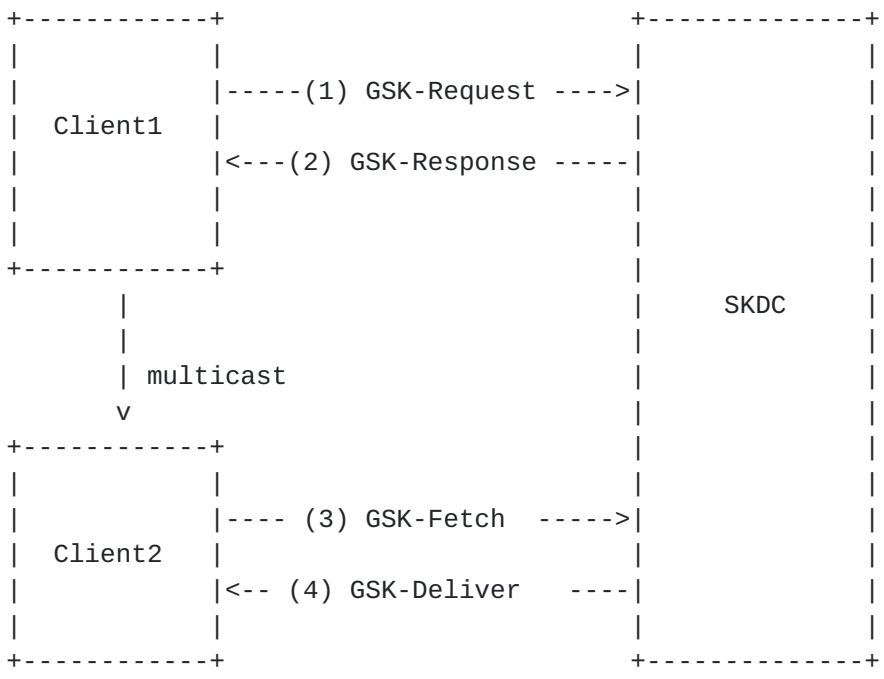

Figure 3

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

- Creation of the GSK at the SKDC:
  - GSK-Request: A client who has a secret key (shared pair-wise with the SKDC) can request the SKDC to create a new group encryption key. The SKDC will retain a copy of the group encryption key (until it expires) and associate it with the multicast group name (mcastname). The client authenticates itself to the SKDC by including an authenticator (encrypted using the client's secret key) in the gsk-request message.
  - GSK-Response: The requesting client obtains a copy of the new group encryption key via the gsk-response message from the SKDC. The gsk-response message uses the receipt structure to carry the group encryption key. The receipt is encrypted using the client's secret key.
- Fetching of a copy of the GSK from the SKDC:
  - GSK-Fetch: A client who has a secret key (shared pair-wise with the SKDC) can request a copy of a group encryption key from the

SKDC. The client must indicate the desired multicast group (mcastname) in the gsk-fetch message. The client authenticates itself to the SKDC by including an authenticator (encrypted using the client's secret key) in the gsk-fetch message.

* GSK-Deliver: The requesting client obtains a copy of the group encryption key via the gsk-deliver message from the SKDC. The gsk-deliver message uses the receipt structure to carry the group encryption key. The receipt is encrypted using the client's secret key.

The message components as used in the protocol are summarized in Figure 4. Note that all protocol messages are integrity-protected, and some are encrypted.

The GSK Message Components

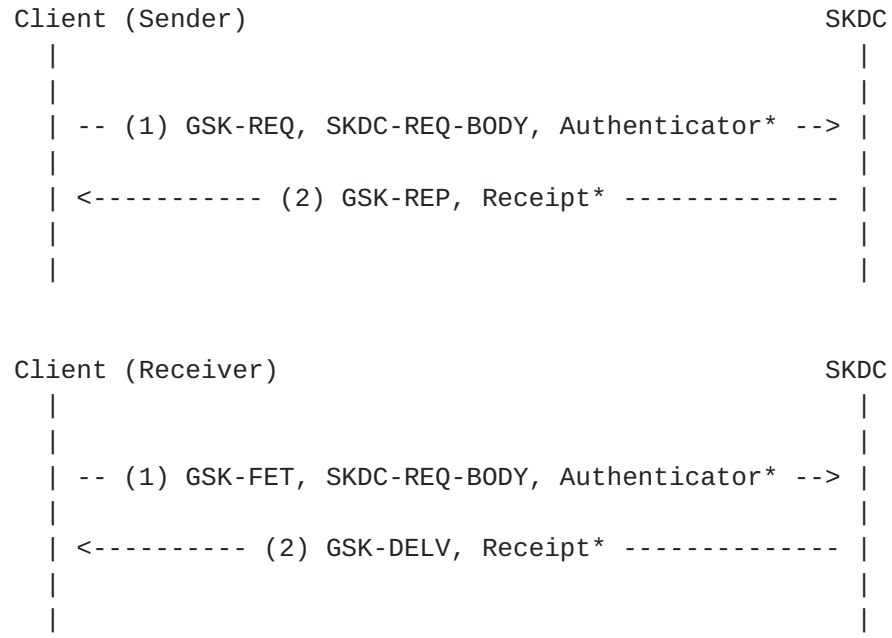

Figure 4

3.1. GSK-Request Message

The GSK-Request message is sent from the client to the SKDC asking the SKDC to create a new group encryption key. The client authenticates itself to the SKDC by including an authenticator (encrypted using the client's secret key) in the gsk-request message.

The contents of the gsk-request 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-REQ".

- SKDC request body: The request body contains the parameters required by the SKDC to create a new group encryption key. In the case of the gsk-request message, the service principal name is instead the desired name of the multicast group (mcastname) to be owned by the client. The SKDC keeps an association between the multicast group name and group encryption key.

  * SKDC options - optional (skdc-options): These are optional flags that the client can set. This field is optional.

  * 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 requesting the new group encryption key.

  * Multicast group identity (mcastname): This is the multicast group name desired by the client.

  * Multicast group realm - optional (mcastrealm): This is the multicast group realm as desired by the client. This field is optional.

  * Client's nonce (nonce): This is the nonce generated by the client.

  * Desired encryption algorithm (etype): This is the encryption algorithm desired/supported by the client for the group encryption key.

- Authenticator: In the case of the gsk-request message, the authenticator is encrypted using the client's secret key shared between the client and the SKDC. See section Section 2.5.1 for the authenticator.

3.2. GSK-Response Message

The GSK-Response message is sent from the SKDC to the client in response to the client's GSK-Request message.

The contents of the GSK-Response 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-REP".
- Client's realm (crealm): This the identity of the realm, domain or group in which the client belongs in connection to this request. This is obtained from the gsk-request message.
- Client's identity (cname): This is the identity of the client obtained from the previous gsk-request message.
- Miniticket (mticket): In the case of the GSK-REP message, miniticket field is empty.
- Receipt (receipt): See Section Section 3.2.1for the receipt structure in the case of the GSK-REP (GSK-DELV) message.

3.2.1. Receipt for GSK

In the case of the GSK-REP and GSK-DELV messages, the receipt contains the following:

- Issuing SKDC's realm (skdcrealm): This is the realm of the SKDC that generated the group encryption key.
- Multicast group identity (mcastname): This is the identity of the multicast group.
- Multicast group realm - optional (mcastrealm): This is the multicast group realm as desired by the client. This field is optional.
- Group encryption key (key): This is the group encryption key generated by the SKDC.
- Time of authentication (authtime): This is the time at the SKDC when it received the gsk-request message.
- Expiration time of key (endtime): The is the expiration time of the group encryption key.
- Nonce from the client's GSK-REQ (GSK-FET) message (nonce): This is the nonce from the client's gsk-request (gsk-fetch) message.
- Group permissions - optional (grp-pac): This is the permissions and access control (PAC) structure related to the multicast group. This field is optional.

- Transited realms - optional (transited): This is the set of SKDC/ realms that was involved in the issuance of the group encryption key. This field is optional.

3.3. GSK-Fetch Message

The GSK-Fetch message is sent by a client to the SKDC asking for a copy of a group encryption key belonging to a group owner. The client must identify the desired multicast group name (mcastname) in the SKDC request body part of the message. The client authenticates itself to the SKDC by including an authenticator (encrypted using the client's secret key) in the gsk-fetch message.

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: The request body in a GSK-FET message has the main purpose of the client identifying the desired multicast group and for the client to deliver a nonce to the SKDC. The SKDC request body contains the following:

  * SKDC options - optional (skdc-options): These are optional flags that the client can set. This field is optional.

  * 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.

  * Multicast group identity (mcastname): This is the identity of the desired multicast group as known by the client.

  * Multicast group realm - optional (mcastrealm): This is the multicast group realm as known by the client. This field is optional.

  * Client's nonce (nonce): This is the nonce generated by the client.

- Authenticator: In the case of the GSK-FET message, this is the data structure encrypted by the client for the SKDC and provides proof of possession to the SKDC of the client secret key. Here the authenticator is encrypted using the secret key shared between

the client and the SKDC. See section Section 2.5.1 for the authenticator.

3.4. GSK-Deliver Message

The GSK-Deliver 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 the client's secret key.

The contents of the GSK-Deliver 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-DELV".
- Client's realm (crealm): This the identity of the realm, domain or group in which the client belongs in connection to this request. This is obtained from the gsk-fetch message.
- Client's identity (cname): This is the identity of the client obtained from the previous gsk-fetch message.
- Miniticket (mticket): In the case of the GSK-DELV message, the miniticket is empty.
- Receipt (receipt): See Section Section 3.2.1 for the receipt structure in the case of the GSK-REP (GSK-DELV) message.

4. JSON Message Format

TBD.

5. Encryption and Checksums

TBD.

6. Security Considerations

TBD.

7. Privacy Considerations

TBD.

8. IANA Considerations

TBD.

9. Acknowledgments

We thank Jesse Walker for design inputs and initial review.

10. References

10.1. 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, March 1997.

[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>.

10.2. 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