Network Working Group
Internet Draft
Category: Informational
Expires: January 17, 2006

N. Cam-Winget
D. McGrew
J. Salowey
H. Zhou
Cisco Sytems
July 17, 2005

Dynamic Provisioning using EAP-FAST
draft-cam-winget-eap-fast-provisioning-01.txt

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Copyright Notice

Abstract

EAP-FAST is an extensible EAP method that enables secure communication between a client and a server by using the Transport Layer Security (TLS) to establish a mutually authenticated tunnel. EAP-FAST also enables the provisioning credentials or other information thru this protected tunnel. This document describes the use of EAP-FAST for dynamic provisioning.

Table of Contents

1. Introduction

[EAP-FAST] is an extensible EAP method that can be used to mutually authenticate peer and server. However, to mutually authenticate with EAP-FAST, credentials such as a preshared key, trusted anchor or a Tunnel PAC MUST be provisioned to the peer before it can establish a secure association with the server. In some cases, the provisioning of such information present deployment hurdles. Through the use of the protected tunnel, EAP-FAST can also be used to enable the means for dynamic or in-band provisioning to address such deployment obstacles.

1.1. Specification Requirements

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].

1.2. Terminology

Much of the terminology in this document comes from [RFC3748]. Additional terms are defined below:

Man in the Middle (MitM)
An adversary that can successfully inject itself between a peer and EAP server. The MitM succeeds by impersonating itself as a valid peer, authenticator or authentication server.

Provisioning
Providing peer with a trust anchor, shared secret or other appropriate information based on which a security association can be established.

Protected Access Credential (PAC)
Credentials distributed to a peer for future optimized network authentication. The PAC consists of at most three components: a shared secret, an opaque element and optionally other information. The shared secret part contains the pre-shared key between the peer and authentication server. The opaque part is provided to the peer and is presented to the authentication server when the peer wishes to obtain access to network resources. Finally, a PAC may optionally include other information that may be useful to the client.

2. EAP-FAST Provisioning Modes

EAP-FAST supports two modes for provisioning:

1) Server-Authenticated Mode: Provisioning inside a server authenticated (TLS) tunnel.

2) Server-Unauthenticated Mode: Provisioning inside an unauthenticated (TLS) tunnel

In the Server-Authenticated Provisioning mode, the peer has successfully authenticated the EAP server as part of the TLS handshake of EAP-FAST Phase 1 (e.g. tunnel establishment). Additional exchanges MAY be needed inside the tunnel for the EAP Server to authenticate the peer before any information can be provisioned.

In the Server-Unauthenticated Provisioning mode, an unauthenticated tunnel is established in the EAP-FAST Phase 1. This provisioning mode is defined to enable bootstrapping or initial configuration of

peers where the peer lacks strong credentials (if any) to mutually authenticate with the server and configuration through out-of-band mechanisms are prohibitive.

In the Server-Unauthenticated Provisioning mode, the peer and server do not achieve mutual authentication during EAP-FAST Phase 1. It is expected that the peer negotiates TLS_DH_anon_WITH_AES_128_CBC_SHA to signal that it can not provide proof of authenticity. While other cipher suites such as those requiring the use of server certificates may be used, the peer may lack the necessary trust anchors to validate the certificate and authenticate the server.

Since the tunnel is not authenticated in the Server-Unauthenticated Provisioning mode, it is possible that the TLS channel may be terminated by an attacker. It is strongly recommended that an inner EAP method be used to provide some authenticity assurances and MitM detection and warning outlined in Section 5 MUST be applied.

The EAP-FAST Phase 2 conversation is unchanged in either Provisioning mode, except that the peer MUST accept an EAP method supporting mutual authentication and key derivation that is compatible with its initial or bootstrapping credentials (such as a password-based EAP method). The peer then uses the Crypto-Binding TLV to validate that the same server terminates both the TLS channel and to successfully complete the EAP method, thereby verifying that the exchange was not subject to a man-in-the-middle attack. Assuming that the Crypto-Binding TLV exchange is successful, the server will subsequently provide the information such as a shared key or the trusted root(s) of server certificate using a PAC TLV or a Server Trusted Root TLV respectively.

Once the EAP-FAST Provisioning conversation completes, the peer is expected to use the provisioned credentials in subsequent EAP-FAST authentications. It is strongly recommended that either or both peer and EAP Server policy enforce Server-Unauthenticated Provisioning mode to be used no more than once as a means to minimize exposure to potential MitM attacks.

3. Dynamic Provisioning using EAP-FAST Conversation

The provisioning EAP-FAST exchange uses same sequence as the EAP-FAST Authentication Phase 1 to establish a protected tunnel by means of a TLS based Diffie-Hellman (DH) key agreement exchange. Once a tunnel

is secured between the two parties, the client and server can then execute an EAP authentication method by which both parties can achieve mutual authentication.

Provisioning in EAP-FAST is negotiated solely by the client as the first communication exchange when EAP-FAST is requested from the server. If the client does not have a Protected Access Credential (PAC) or requires provisioning of other information (such as the serverÆs Trusted Root certificate), it can request to initiate a provisioning EAP-FAST exchange and dynamically obtain one from the server.

The EAP-FAST provisioning conversation will typically occur between the peer and an authentication server; more specifically, the server that can provision the peer with the requested information; typically, a unique PAC.

The conversation between a peer and authentication server commences as a normal EAP-FAST exchange: with an anonymous Identity for a peer and the server determining that EAP-FAST authentication is to occur, the EAP server MUST respond with an EAP-FAST/Start packet. Assuming that the peer supports EAP-FAST and the peer has no PAC provisioned on its device, the peer shall send an EAP-Response packet with EAP-Type=EAP-FAST.

On receipt of the EAP-FAST Start message, the peer determines it must be provisioned with a fresh PAC. Further, the peer determines whether it must invoke a signed or anonymous DH exchange.

When an anonymous key exchange is negotiated, the signature in the KeyExchange algorithm shall contain the sha_hash of the records as defined in [RFC 2246]. If a signed key exchange is negotiated, then the DH parameters are signed using the serverÆs private key; with a signed key exchange the server may also include a certificate to enable the peer to validate the signature.

To provide best security practices, it is highly recommended that the peer obtain the serverÆs public key or trust anchor to enable server-side authentication by employing a signed Diffie-Hellman exchange (e.g. TLS_DHE_RSA_WITH_AES_128_CBC_SHA cipher suite specification). However, as the provisioning of the public key or trust anchor must also be secured to ensure the public key is to be trusted, some deployments may be willing to trade off the security risks for ease of deployment.

The peer and server establish the EAP-FAST tunnel for provisioning in the same exchanges as that defined for EAP-FAST authentication [EAP-FAST]. With a successful EAP-FAST Phase 1 tunnel established, subsequent messages exchanged between peer and authentication server are protected using 128bit AES in CBC mode and HMAC-SHA1 as defined by both [RFC 2246] and [RFC 3268] to provide message confidentiality and integrity respectively.

With a protected tunnel, the peer must now authenticate itself to the server before the server can provision it with a PAC. To ensure some means for authentication and to protect such authentication from exposure, the provisioning EAP-FAST exchange also employs [MSCHAPv2] to achieve mutual authentication before any credentials or information can be provisioned. If an anonymous DH exchange ensued to establish the tunnel or if the peer was unable to validate the authenticated DH exchange, the MSCHAPv2 exchange is susceptible to an active server-side dictionary attack. However, as it enables in-band provisioning at the cost of some loss in security strength, it is an option to afford a means for facilitating a deployment with minimal to no client (peer) configuration. To minimize exposure of the active dictionary attack, it is recommended that the anonymous DH provisioning EAP-FAST conversation be used only once; further provisioning or updates of the PAC should be done by means of the EAP-FAST PAC refreshing protocol or through some other (manual or out-of-band) mechanisms.

The client authentication proceeds by the peer and authentication server engaging in an MSCHAPv2 conversation using invoking the same EAP-FAST Phase 2 MSCHAPv2 conversation. To further mitigate man-in-the-middle attacks in the Server-Unauthenticated Provisioning Mode, the challenges provided by the peer and authentication server are generated as part of the TLS establishment in the EAP-FAST provisioning exchange and conveyed as the Server and Client Challenges requested by MSCHAPv2. Further, the random challenges are not conveyed in the actual MSCHAPv2 messages, the messages shall replace the fields with zeroes to obscure the actual values used to generate the challenge responses.

Following a successful MSCHAPv2 authentication exchange and successful Intermediate Result TLV and Crypto-Binding TLV exchange, the server can then provision the peer with a unique PAC. The provisioning is invoked through the same mechanism as in PAC refreshment: a PAC-TLV exchange is executed following the successful

MSCHAPv2 exchange including the Intermediate Result TLV and Crypto-Binding TLV exchange, with the server distributing the PAC in a corresponding PAC TLV to the peer and the peer confirming its receipt in a final PAC TLV Acknowledgement message.

3.1 Network Access after EAP-FAST Provisioning

Depending on server policy, network access can be granted or denied based on the EAP-FAST Provisioning mode, the credential(s) or other information that have been provisioned, and the inner EAP methods used. For example, in the Server-Authenticated Provisioning Mode, access can be granted after the EAP server has authenticated the peer and provisioned the peer with a Tunnel PAC (e.g. a PAC used to mutually authenticate the EAP-FAST tunnel).

Additionally, peer policy may also be used to disconnect the current provisioning connection and initiate a new EAP-FAST exchange for authentication utilizing the newly provisioned information and ensure the inner methods are conducted with the trusted server. The peer policy may be required as the peer determines whether it can authenticate the EAP Server. In the case where a peer lacks the trust anchors to validate the serverÆs certificate, the peer SHOULD negotiate the TLS_DH_anon_WITH_AES_128_CBC_SHA to signal the EAP server that it lacks the trust anchors to authenticate the server.

At the end of the Server-Unauthenticated Provisioning Mode, network access SHOULD NOT be granted. EAP server SHOULD conclude with an EAP Failure to acknowledge that this conversation was intended for provisioning only and thus no network access is authorized. Upon completion of the exchange, the EAP Server SHALL NOT grant network access or distribute any session keys to the NAS as this phase is not intended to provide network access. Even though provisioning mode completes with a successful inner termination (e.g. successful Result TLV), server policy defines whether the peer gains network access or not. Thus, it is feasible for the server, while providing a successful Result TLV may conclude with an EAP Failure.

The EAP-FAST server, when denying network access after EAP-FAST Provisioning, may choose to instead, immediately invoke another EAP-FAST Start and thus initiate the EAP-FAST Phase 1 conversation. This server based implementation policy may be chosen to avoid applications such as wireless devices from being disrupted (e.g. in

802.11 devices, an EAP Failure may trigger a full 802.11 disassociation) and allow them to smoothly transit to the subsequent EAP-FAST authentications to enable network access.

Similarly, if Server-Authenticated Provisioning Mode is used and the server policy is to disallow network access, the EAP Server SHALL NOT grant network access or distribute any session keys to the NAS as this phase is not intended to provide network access. Even though provisioning mode completes with a successful inner termination (e.g. successful Result TLV), the EAP-FAST Server-Authenticated Provisioning Mode MUST conclude with an EAP Failure to acknowledge that this conversation was intended for provisioning only and thus no network access is authorized. The EAP-FAST server may choose to instead, immediately invoke another EAP authentication transaction.

3.2 Authenticating Using MSCHAPv2

While other authentication methods exist to achieve mutual authentication, when using an anonymous or unauthenticated TLS tunnel, MSCHAPv2 was chosen for several reasons:

* Afford the ability of slowing an active attack by obscuring the password through some hash

* Especially in the Server-Unauthenticated EAP-FAST Provisioning conversation MSCHAPv2 affords the ability to detect, based on the challenge responses, whether there is a possible attack.

* It is understood that a large deployed base is already able to support MSCHAPv2

* MSCHAPv2 is picked in order to slow the active dictionary attack relative to MSCHAPv1.

* Allow support for password change during the EAP-FAST Provisioning protocol.

The MSCHAPv2 exchange forces the server to provide a valid ServerChallengeResponse which must be a function of the server challenge, client challenge and password as part of its response. This reduces the window of vulnerability in the EAP-FAST for in-band provisioning protocol to force the man-in-the-middle, acting as the

server, to successfully break the password within the clientÆs challenge response time limit.

EAP-FAST for provisioning only specifies MSCHAPV2 as the inner method when using an anonymous DH key agreement. However, with support of signed DH key agreement, the provisioning protocol of EAP-FAST may support other methods such as EAP-GTC to enable peers (using other password databases such as LDAP and OTP) to be provisioned in-band as well. However, the replacement may only be achieved when used with the TLS_DHE_RSA_WITH_AES_128_CBC_SHA cipher suite to ensure no loss in security.

When using an anonymous DH key agreement and MSCHAPv2, a binding between the tunnel and the MSCHAPv2 exchanges is formed by using keying material generated during the EAP-FAST tunnel establishment as the MSCHAPv2 challenges. A detailed description of the challenge generation is described in Section 3.4.

3.3 Use of other Inner EAP Methods for EAP-FAST Provisioning

Once a protected tunnel is established, the peer must authenticate itself to the server before the server can provision the peer. When using TLS_DH_anon_WITH_AES_128_CBC_SHA cipher suite in the EAP-FAST Phase 1 conversation, EAP-MSCHAPv2 is the only inner method allowed for Dynamic Provisioning in EAP-FAST.

The MSCHAPv2 exchange forces the server to provide a valid ServerChallengeResponse which must be a function of the server challenge, client challenge and password as part of its response. This reduces the window of vulnerability in this provisioning mode to force the man-in-the-middle, acting as the server, to successfully break the password within the clientÆs challenge response time limit, or it raises the ability to detect if an MitM attacker is capturing the MSCHAPv2 exchange without responding for the purpose of affecting an off-line dictionary attack on the password.

As a result of not authenticating the server in Phase 1 and potential MITM attacks in the Server-Unauthenticated Provisioning Mode, an EAP method with equal or better protection than EAP-MSCHAPv2 MUST be used in Phase 2.

With the use of additional TLS cipher suites, especially when serverauthenticity is verified as part of the TLS tunnel

establishment, other inner EAP methods with weaker protection than EAP-MSCHAPv2 can be used safely inside tunnel. Hence, in addition to EAP-MSCHAPV2 as the inner method, EAP-GTC MAY be used in Server-Authenticated Provisioning Mode. This will enable peers using other user databases such as LDAP and OTP to be provisioned in-band as well. However, the replacement may only be achieved when used with the TLS cipher suites that ensure server authentication, such as TLS_DHE_RSA_WITH_AES_128_CBC_SHA, to ensure no loss in security.

Dynamic Provisioning EAP-FAST MUST support both EAP-GTC and EAP-MS-CHAPv2 within the tunnel in Server-Authenticated Provisioning Mode.

It should be noted that Server-Authenticated Provisioning Mode provides significant security advantages over Server-Unauthenticated Provisioning even when EAP-MSCHAPv2 is being used as inner method. It protects the EAP-MSCHAPv2 exchanges from potential MitM attacks by verifying serverÆs authenticity before exchanging MSCHAPv2. Thus Server-Authenticated Provisioning Mode is the preferred provisioning mode. The EAP-FAST peer MUST use the Server-Authenticated Provisioning Mode whenever a certificate or (serverÆs) public key is available to authenticate the server, in order to ensure best security practices.

3.4 Key Derivations Used in the EAP-FAST Provisioning Exchange

When generating keys in the EAP-FAST Provisioning conversation, the DH computation is used as the pre_master_secret and is converted into the master_secret as specified by [RFC 2246]:

For the client:
  pre_master_secret = (DH_Ys)^peer-private-DH-key mod DH_p

For the server:
  pre_master_secret = (DH_Yc)^server-private-DH-key mod DH_

master_secret = PRF(pre_master_secret, ômaster secretö, client_random + server_random)

The TLS tunnel key is calculated similar to the TLS key calculation with an extra 72 octets generated. Portions of the extra 72 octets are used for the EAP-FAST provisioning exchange session key seed and as the random challenges in the MSCHAPv2 exchange.

To generate the key material, compute

```
key_block = PRF(master_secret,
                    ôkey expansionö,
                    server_random +
                    client_random);
```

until enough output has been generated. Then the key_block is partitioned as follows:

```
client_write_MAC_secret[hash_size]
server_write_MAC_secret[hash_size]
client_write_key[Key_material_length]
server_write_key[key_material_length]
client_write_IV[IV_size]
server_write_IV[IV_size]
session_key_seed[seed_size= 40]
MSCHAPv2 ServerChallenge[16]
MSCHAPv2 ClientChallenge[16]
```

The extra key material, session_key_seed is used for the Crypto-Binding while the ServerChallenge and ClientChallenge correspond to the authentication serverÆs MSCHAPv2 challenge and the peerÆs MSCHAPv2 challenge respectively. The ServerChallenge and ClientChallenge are only used for the MSCHAPv2 exchange when TLS_DH_anon_WITH_AES_128_CBC_SHA is used in the EAP-FAST tunnel establishment.

3.5 Provisioning or Refreshment of a PAC

The server may provision or refresh information by use of the Protected Access Credential (PAC) after a successful user authentication. A PAC TLV is defined to facilitate the distribution and refreshing of information and is defined in Section 4.2. A fresh PAC may be distributed after a successful Intermediate Result TLV and Crypto-Binding TLV exchange, if the server detects that the PAC is expiring soon. A successful EAP-FAST inner method authentication, including a successful Crypto-Binding exchange must ensue before an EAP-FAST server can distribute a fresh PAC. A PAC TLV should not be accepted if it is not TLS tunnel-encapsulated.

N.B. In-band PAC refreshing is enforced by server policy. The server, based on the PAC-Opaque information, may determine not to refresh a peerÆs PAC through the PAC TLV mechanism even if the PAC-Key has expired.

4. Types of Information Provisioned in EAP-FAST

In addition to the Tunnel PAC (the one used to establish the EAP-FAST Phase 1 tunnel), other types of credentials and information can also be provisioned. They may include trusted root certificates for the server certificates, application specific PACs, and user identities to name a few. Typically, provisioning is invoked after both peer and server validate their authenticities and after a successful Crypto-Binding TLV exchange. However, depending on the information being provisioned, mutual authentication MAY not be needed.

At minimum, at least one entity (peer or server) must prove authenticity before credentials are provisioned to ensure that information is not freely provisioned to or by adversaries. For example, the EAP server MAY not need to authenticate the peer to provision the peer with trusted root certificates. However, the peer MUST authenticate the server before it can accept a trusted server root certificate.

The server distributes all PAC information through the use of a PAC TLV. Each type of PAC information is typed through a PAC Type and PAC TLV Attribute defined in this section.

4.1 PAC Types

A Protected Access Credential (PAC) is a security credential provided by the Authentication Server (AS) that holds application specific information. For instance, a Tunnel PAC holds a shared secret mutually and uniquely shared between the peer and AS and is used to secure an EAP-FAST (TLS) tunnel. EAP-FAST uses the PAC to facilitate the storage of secure information between a peer and a server on the peer and minimize the per user state management on the AS.

However, as PACs have wider contextual use, the PAC used for establishing the EAP-FAST tunnel in Phase 1 is referred to as the Tunnel PAC throughout this document. A summary of three major types

of PACs that MUST be supported in this version of the Dynamic Provisioning EAP-FAST include:

1) Tunnel PAC û A distributed shared secret between the peer and AS used to establish a secure the EAP-FAST tunnel and convey the server policy of what must and can occur in the tunnel. The server policy can include EAP methods, TLV exchanges and identities allowed in the tunnel. It is up to the server policy to include whatÆs necessary in a PAC to enforce the policy in subsequent authentications that use the PAC. For example, user identity, I-ID, can be included as the part of the server policy. This I-ID information limits the inner EAP methods to be carried only on the specified user identity. Other types of information can also be included, such as which EAP method(s) and which cipher suite is allowed. If the server policy is not included in a PAC, then there is no validation or limitation on the inner EAP methods or user identities inside the tunnel established by use of that PAC.

2) Application Specific Short Lived PACs - these PACs carry authorization information that is bound to a specific user or device identity. They are intended to be short-lived, with an expiry time usually in the range of normal session resume timeouts (i.e., minutes or hours, versus days or months). Since they are usually bound to a particular session or state, they MUST be kept in volatile memory only and MUST not be persistent cross system reboots. They MAY be bound to the Tunnel PAC to enforce the two being used together. Currently there is only one application specific short lived PAC defined as:

   User Authorization PAC û A distributed user authentication and authorization result based on a previous authentication. It can be used in combination with the Tunnel PAC to bypass subsequent user authentication(s). It is intended to be short-lived and also controlled by the peer. If any state of the user has changed to the extent that will affect the user authentication result (i.e., user has logged on/off), the peer MUST discard it and not use it again. The User Authorization PACs can be used in combination with the Tunnel PAC to allow a stateless server session resume as defined [EAP-FAST].

3) Application Specific Long Lived PACs - Application specific long lived PACs are each issued a key that can be used to prove ownership of the PACs and credentials. They can be considered

another form of credentials, independent of the Tunnel PAC and can be used alone to prove authenticity. In this case they may not be bound to the Tunnel PAC. They are usually allowed a longer expiry time and stored in persistent storage. Peer and EAP Server can use them either inside or outside EAP-FAST (mostly likely in some shared key EAP methods) to manually authenticate each other. One application of this type of PAC is defined in this specification:

Machine Authentication PAC û A distributed device authentication and authorization policy based on a previous user authentication. It can be used by itself to prove ownership of the PAC and gain authorization. It is often used as the credentials to obtain network access in lieu of the user credentials whenever user credentials are not available, i.e., during boot up time. After successful validation of the Machine Authentication PAC, limited or full network access MAY be granted based on the serverÆs policy.

To request provisioning of a Tunnel PAC, a peer MUST send a PAC TLV with a PAC-Type PAC TLV with its TLVs field and set to ô1ö (Tunnel PAC Type). The request may be issued after the peer has determined that it has successfully authenticated the EAP Server and the tunnel and inner EAP methods were between the same peer and EAP Server by validating the Crypto-Binding TLV. This would differentiate the Tunnel PAC request from other types of PAC provisioning requests. If anonymous DH is negotiated and the peer does not send any PAC-TLV to request provisioning, then Tunnel PAC is provisioned automatically by the server. PAC-Acknowledge TLV MUST be used for peer to acknowledge the receipt of the Tunnel PAC.

To request provisioning of PACs other than the Tunnel PAC, a peer MUST send a PAC TLV with a PAC-Type PAC TLV in its TLVs field and set to the appropriate PAC-Type, after the peer has determined that it has successfully authenticated the EAP Server and determined that the tunnel and inner EAP methods were between the same peer and EAP Server by validating the Crypto-Binding TLV. This can also be used to indicate a peerÆs support for other types of PACs. Peer MUST send each individual corresponding PAC TLV to request different types of PACs. Multiple PAC TLVs can be sent in the same packet or different packets to request provisioning of different type of PACs. The EAP server will send the PACs after its internal policy has been satisfied; or it may ignore the request or request additional

authentications if its policy dictates. If a peer receives a PAC with unknown type, it MUST ignore it.

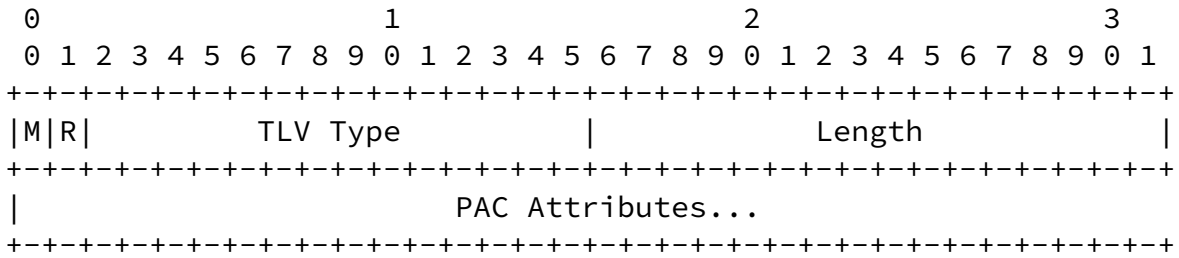
PAC-Acknowledge TLV MUST NOT be used from peer to acknowledge the receipt of other types of PACs.

Please see Section 10.5 for an example of packet exchanges to provision a User Authorization PAC.

4.2 Provisioning PACs through PAC TLV

The PAC TLV is defined to enable the provisioning of PAC information. Additionally, the PAC-Type, PAC TLV MAY be used by the peer to request provisioning for specific types of information. Conversely, the PAC TLV is used by the server to provision the requested information to a peer.

The PAC TLV provides support for Protected Access Credential (PAC) defined within [EAP-FAST]. A consistent PAC format will allow it to be used by multiple applications beyond EAP-FAST. A general PAC TLV format is defined as follows:



M

0 - Non-mandatory TLV
1 - Mandatory TLV

R

Reserved, set to zero (0)

TLV Type

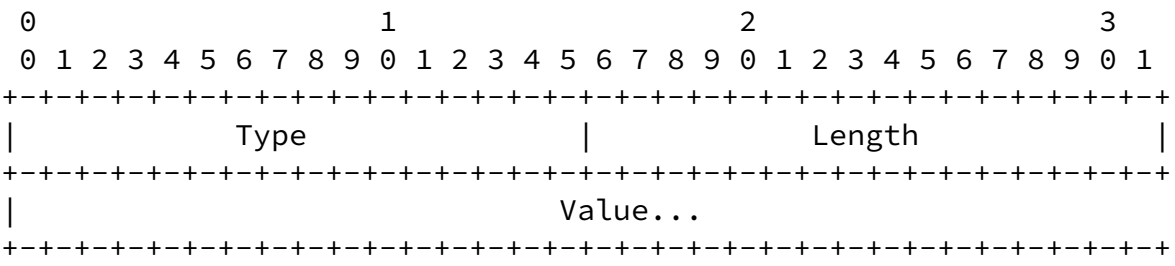
11

Length
The length of the PAC Attributes field in octets.

PAC Attributes
A list of PAC attributes in the TLV format.

4.2.1 Formats for PAC TLV Attributes

A common encapsulating format is used to convey the different fields that comprise a PAC attribute. The common encapsulation is defined as follows:



Type
The type field is two octets, denoting the attribute type. Allocated Types include:

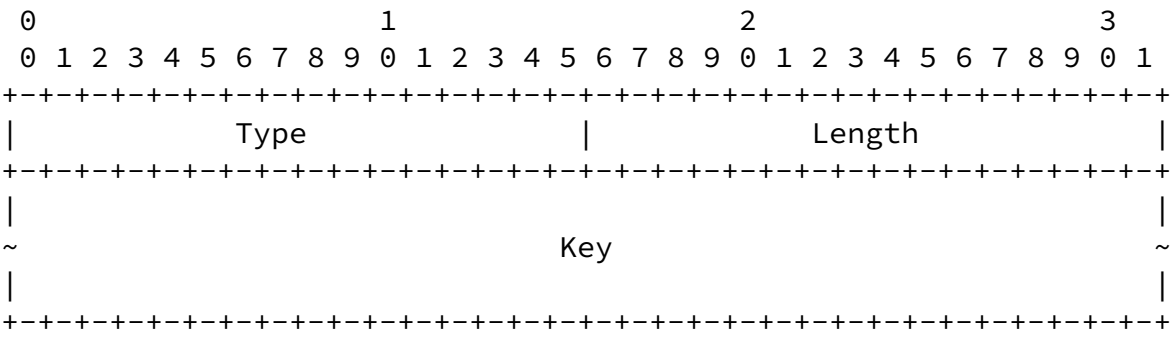
1 - PAC-Key
2 - PAC-Opaque
3 - CRED_LIFETIME
4 - A-ID
5 - I-ID
6 - SERVER_PROTECTED_DATA
7 - A-ID-Info
8 - PAC-Acknowledgement
9 - PAC-Info
10 - PAC-Type

Length
The Length filed is two octets, which contains the length of the Value field in octets.

Value
The value of the PAC Attribute.

4.2.2 PAC-Key

The PAC-Key is distributed as an attribute of type PAC-Key (Type=1). The key is a randomly generated octet string. The key is represented as an octet string whose length is determined by the length field. The generator of this key is the issuer of the credential, identified by the A-ID.



Type

1 - PAC-Key

Length

The Length filed is two octets. For this version of EAP-FAST, PAC-Key is 32 octets.

Key

The Key field contains the PAC-Key.

4.2.3 PAC-Opaque

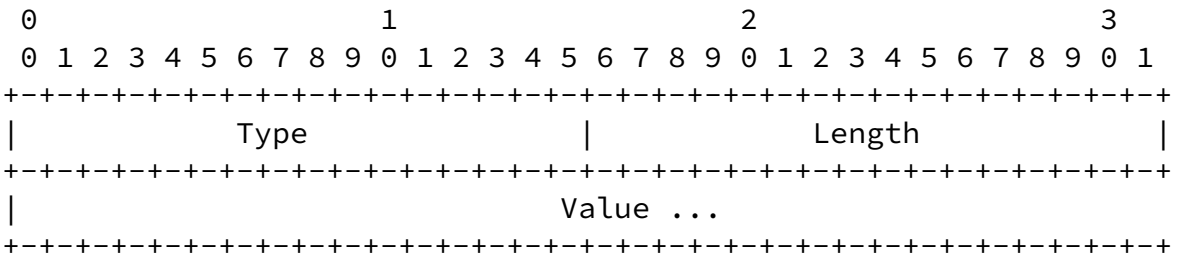
The PAC-Opaque contains data that is opaque to the recipient, the peer is not the intended consumer of PAC-Opaque and thus should not attempt to interpret it. A peer that has been issued a PAC-Opaque by a server MUST store that data, and present it back to the server as is, in the ClientHello SessionTicket extension field [TICKET]. If a client has opaque data issued to it by multiple servers, then it MUST store the data issued by each server separately. This requirement

allows the client to maintain and use each opaque data as an independent PAC pairing, with a PAC-Key mapping to a PAC-Opaque identified by the A-ID. As there is a one to one correspondence between PAC-Key and PAC-Opaque, the peer must determine the PAC-Key and corresponding PAC-Opaque based on the A-ID provided in the EAP-FAST/Start message and the A-ID provided in the PAC-Info when it was provisioned with a PAC-Opaque.

As the PAC-Opaque is server specific, its contents and definition are specific to the issuer of the PAC, e.g. the PAC server.

The PAC-Opaque field is embedded as part of the PAC TLV when the server has determined that the PAC must be refreshed and sends a new PAC.

The PAC-Opaque field format is summarized as follows:



Type

2 - PAC-Opaque

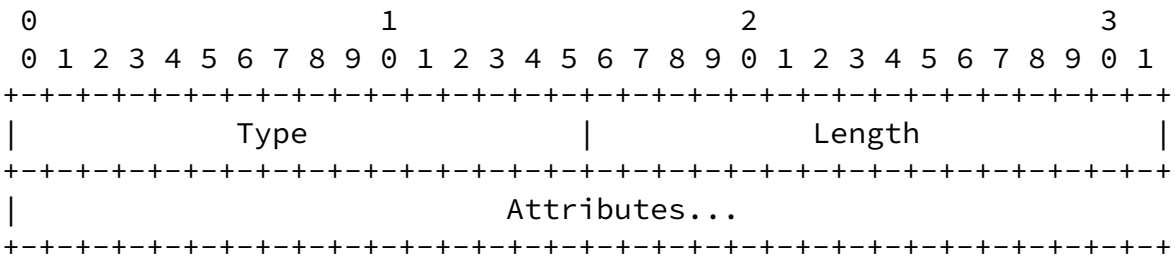
Length
The Length filed is two octets, which contains the length of the value field in octets.

Value
The Value field contains the actual data for PAC-Opaque

The PAC-Opaque field is also passed from the peer to the server during the EAP-FAST Authentication Phase 1 conversation to enable the server to validate and recreate the PAC-Key. When it is passed from the peer, it is encapsulated as defined above in the ClientHello SessionTicket Extension [TICKET].

4.2.4 PAC-Info

PAC-Info is comprised of a set of PAC attributes. At minimum, the A-ID TLV is required to convey the issuing identity to the peer. Other optional fields may be included in the PAC to provide more information to the peer. It is a container attribute for various types of information each of which is encoded in conformance to the PAC TLV field format as defined in Section 4.2.



Type

9 - PAC-Info

Length
The Length filed is two octets, which contains the length of the Attributes field in octets.

Attributes
The Attributes field contains a list of PAC Attributes.

Each mandatory and optional field type is defined as follows:

CRED_LIFETIME (type 3)
This is a 4 octet quantity representing the expiration time of the credential in UNIX UTC time. This is a mandatory field contained in the PAC-Opaque field to enable the server to validate the PAC. This field may also be optionally provided to the peer as part of PAC-Info.

A- ID (type 4)
Authority identifier is the name of the authority that issued the token. The A-ID is intended to be unique across all issuing servers to avoid namespace collisions. Server implementations should use measures to ensure the A-ID used

is globally unique to avoid name collisions. The A-ID is used by the peer to determine which PAC to employ. Similarly, the server uses the A-ID to both authenticate the PAC-Opaque and determine which master key was used to issue the PAC. This field is mandatory and included in both the PAC-Opaque and as the first TLV comprising PAC-Info.

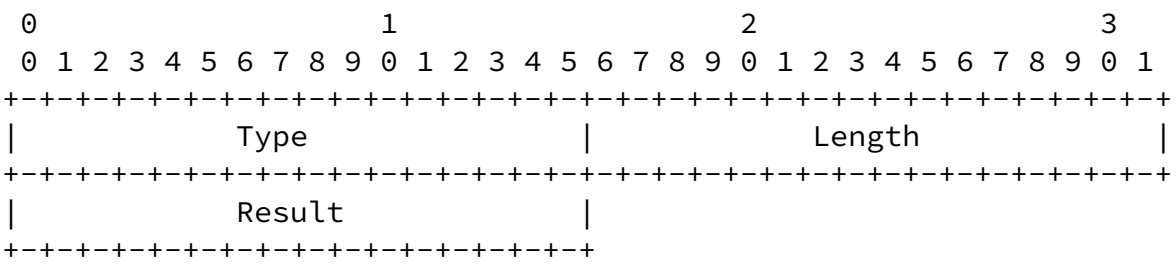
I-ID (type 5)
Initiator identifier (I-ID) is the peer identity associated with the credential. The server employs the I-ID in the EAP-FAST Phase 2 conversation to validate that the same peer identity used to execute EAP-FAST Phase 1 is also used in at minimum one inner EAP method in EAP-FAST Phase 2. This field is a mandatory field in PAC-Opaque and may optionally be included in the PAC-Info. If the AS is enforcing the I-ID validation on inner EAP method, then I-ID is mandatory in PAC-Info, to enable the client to also enforce a unique PAC for each unique user. If I-ID is missing from the PAC-Info, it is assumed that the Tunnel PAC can be used for multiple users and client will not enforce the unique Tunnel PAC per user policy.

A-ID-Info (type 7)
Authority Identifier Information is a mandatory TLV intended to provide a user-friendly name for the A-ID. It may contain the enterprise name and server name in a more human-readable format. This TLV serves as an aid to the peer to better inform the end-user about the A-ID. This field is a mandatory field in the PAC-Info.

PAC-Type (type 10)
PAC-Type is a mandatory TLV intended to provide the type of PAC. This field is a mandatory field in the PAC-Info. If PAC-Type is not present, then it defaults to a Tunnel PAC (Type 1).

4.2.5 PAC-Acknowledgement TLV

The PAC-Acknowledgement TLV is used to acknowledge the receipt of the Tunnel PAC by the peer. Peer sends this TLV in response to the PAC TLV to indicate the result of the retrieving and storing of the new Tunnel PAC. This TLV is only used when provisioning Tunnel PAC.



Type

8 - PAC-Acknowledgement

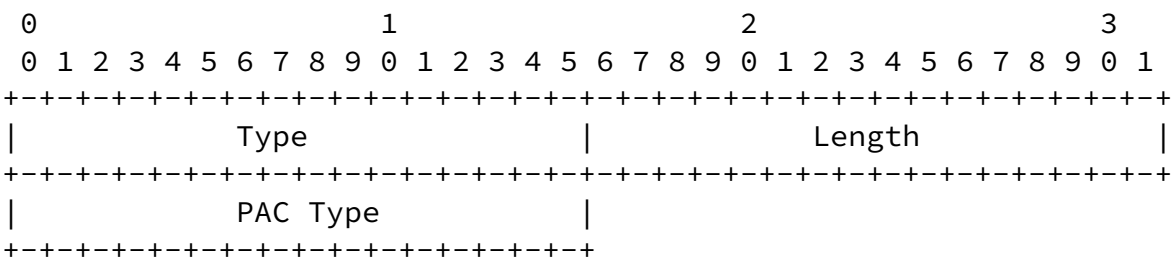
Length
The length of this field is two octets and value must be 2.

Result
The resulting value must be one of the following:

1 - Success
2 - Failure

4.2.6 PAC-Type TLV

The PAC-Type TLV is a mandatory TLV intended to specify the PAC type. Its format is described below.



Type

10 - PAC-Type

Length
The length of this field is two octets and value must be 2.

PAC Type
This two octet field defined the type of PAC being requested or provisioned. Its value must be one of the following:

1 û Tunnel PAC
2 û Machine Authentication PAC
3 û User Authorization PAC

4.3 Server Trusted Root Certificate

It is desirable to provision the peer with the serverÆs trusted root certificates (or CA certificates), which can later be used for enabling PKI based cipher suites. Server-Trusted-Root TLV [EAP-FAST] is introduced to facilitate the request for and delivery of server trusted root certificates. Within the EAP-FAST Phase 2 conversation, a peer MAY request for a serverÆs trusted root certificate using a Server-Trusted-Root TLV, and the EAP server MAY respond with a Server-Trusted-Root TLV containing the trusted root certificate in the PCKS#7 TLV to the peer. The Server-Trusted-Root TLV can be exchanged in regular EAP-FAST Authentication mode or Provisioning modes.

After the peer has determined that it has successfully authenticated the EAP server and determined that the tunnel and inner EAP methods were between the same peer and EAP Server by validating the Crypto-Binding TLV, it MAY send one or more Server-Trusted-Root TLVs (marked as optional) to request for the certificate trust anchors of the server certificate from the EAP server. The EAP server will send the trusted root(s) of server certificate after its internal policy has been satisfied; or it may ignore the request or request additional authentications based on its policy. The peer may receive a trusted root of server certificate, but is not required to use it. Please see Section 10.4 for an example of a server provisioning a server trusted root certificate.

4.3.1 Server-Trusted-Root TLV

The Server-Trusted-Root TLV allows the peer to send a request to the EAP server for a trusted root in PKCS#7 format.

The Server-Trusted-Root TLV is always marked as optional, and cannot be responded to with a NAK TLV.

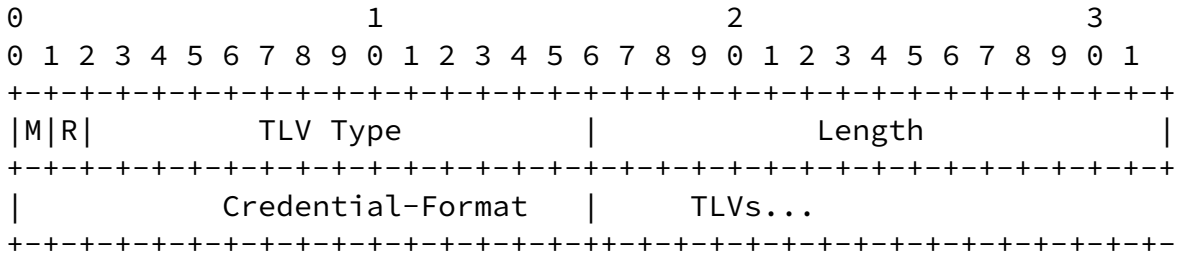
The Server-Trusted-Root TLV can only be sent as an inner TLV (inside the protection of the tunnel).

The peer MUST NOT request, or accept the trusted root sent inside the Server-Root credential TLV by EAP-server until it has completed authentication of EAP server, and validated the Crypto-Binding TLV. The peer may receive a trusted root, but is not required to use the trusted root received from the EAP server.

If the EAP server sets credential-format to PKCS#7-Server-Certificate-Root, then the Server-Trusted-Root TLV MUST contain the root of the certificate chain of the certificate issued to the EAP server packages in a PKCS#7 TLV. If the Server certificate is a self-signed certificate, then the root is the self-signed certificate. In this case, the EAP server does not have to sign the certificate inside the PCKS#7 TLV since it does not necessarily have to private key for it.

If the Server-Trusted-Root TLV credential format does not contain one of the known values, then the EAP-server MUST ignore the value.

The Server-Trusted-Root TLV is defined as follows:



M
0 - Optional TLV

R
Reserved, set to zero (0)

TLV Type
18

Length
>=2

Credential-Format
The Credential-Format field is two octets. Values include:

1 - PKCS#7-Server-Certificate-Root.

TLVs
This field is of indefinite length. It contains TLVs associated with the certificate-request.

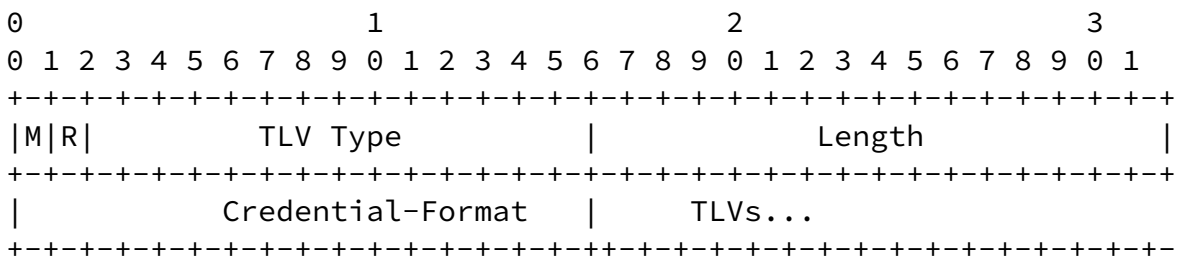
4.3.2 PKCS #7 TLV

The PKCS#7 TLV is sent by the EAP server to the peer inside the Server-Trusted-Root TLV. It contains the PKCS #7 wrapped X.509 certificate. This field contains a certificate or certificate chain in PKCS#7 format requested by the peer as defined in [RFC2315].

The PKCS#7 TLV is always marked as optional, which cannot be responded to with a NAK TLV. EAP-FAST server implementations that claim to support provisioning MUST support this TLV. EAP-FAST peer implementations MAY not support this TLV.

If the PKCS#7 TLV contains a certificate or certificate chain that is not acceptable to the peer, then peer MUST ignore the value.

The PKCS#7 TLV is defined as follows:



M
0 - Optional TLV

R
   Reserved, set to zero (0)

TLV Type
   20 (for PKCS #7 TLV)

Length
   The length of the PKCS #7 Data field

PKCS #7 Data
   This field contains the PKCS #7 wrapped X.509 certificate or certificate chain in the PKCS #7 format.

5. Security Considerations

The Dynamic Provisioning EAP-FAST protocol shares the same security considerations outlined in [EAP-FAST]. Additionally, it also has its unique security considerations described below:

5.1 User Identity Protection and Validation

EAP-FAST for provisioning employs the DH key agreement (as defined in the TLS protocol) to establish a protected tunnel; the initial Identity request/response may be omitted as it must be transmitted in the clear and thus subject to snooping and forgery. Alternately, an anonymous identity may be used in the Identity response to prevent disclosure of the peerÆs true identity.

As the provisioning EAP-FAST exchange is used for provisioning a PAC to a specific identity, e.g. I-ID, it is expected that the server will assign the I-ID based on the identity provided in the protected inner EAP authentication method. Thus, the protected identity may not be identical to the cleartext identity presented in the initial tunnel establishment messages. In order to shield the user identity from snooping, the cleartext Identity may only provide enough information to enable routing of the authentication request to the correct realm. For example, the peer may initially claim the identity of "nouser@bigco.com" in order to route the authentication request to the bigco.com EAP server. Subsequently, once the EAP-FAST session has

been negotiated, in the inner authentication method, the peer may claim the identity of "fred@bigco.com". Thus, the EAP-FAST protocol for provisioning can provide protection for the user's identity, though not necessarily the destination realm, unless the provisioning EAP-FAST conversation terminates at the local authentication server.

5.2 Mitigation of Dictionary Attacks

When EAP-FAST is invoked for provisioning, the peer specifies the means for securing the communications for the provisioning. As such, it can invoke the DH key agreement in one of two ways: anonymously or server-authenticated. With a server-authenticated DH key agreement, the server must provide its certificate and an RSA signature with the ephemeral DH parameters, whereas no signature is provided for an anonymous DH key agreement.

In a server authenticated DH key agreement, the protected communications is assured that the AS is authentic as the peer must have been pre-provisioned with the ASÆs certificate or public (RSA) key prior to the negotiation. As it is the peer that must first provide proof of identity through an identity and (password) credential, an adversary may only pose as an AS to successfully mount a dictionary attack. An EAP-FAST compliant implementation must assure that provisioning of the AS public key, certificate or root certificate to the peer must be achieved through a secure mechanism. Only through a secure mechanism can server-authenticated DH key agreement provide resistance to dictionary attacks. While this option affords best security practices, it presents deployment issues as, especially for wireless clients where there is little means to provide secure configuration, peers must be configured with a means to validate the serverÆs credential (e.g. public key).

In an anonymous DH key agreement, an adversary may attempt to impersonate a client and enable EAP-FAST for provisioning. However, it must successfully authenticate inside the DH tunnel to succeed and gain a PAC credential from a server. Thus, peer impersonation is mitigated through the enabling of peer authentication inside a protected tunnel. However, an adversary may impersonate as a valid AS and gain the peerÆs identity and credentials. While the adversary must successfully gain contact with a peer that is willing to negotiate EAP-FAST for provisioning and provide a valid A-ID that a client accepts, this occurrence is feasible and enables an adversary to mount a dictionary attack. For this reason, an EAP-FAST compliant implementation must support an MSCHAPv2 or stronger EAP

method for peer authentication when an anonymous DH key agreement is used for the tunnel establishment.

With MSCHAPv2, a peer may detect it is under attack when the AS that has provided an acceptable Authority ID (A-ID) fails to provide a successful MSCHAPv2 server challenge response. By employing the ServerChallenge and ClientChallenge derived during tunnel establishment; detection of a MitM is feasible during the MSCHAPv2 exchange.

The peer may choose to use a more secure out-of-band mechanism for PAC provisioning that affords better security than the anonymous DH key agreement. Similarly, the peer may find a means of pre-provisioning the serverÆs public key securely to invoke the server-authenticated DH key agreement.

The anonymous DH key agreement is presented as a viable option as there may be deployments that can physically confine devices during the provisioning or are willing to accept the risk of an active dictionary attack. Further, it is the only option that enables zero out-of-band provisioning and facilitates simpler deployments requiring little to no peer configuration.

5.3 Mitigation of Man-in-the-middle (MitM) attacks

EAP-FAST invocation of provisioning addresses MitM attacks in the following way:

* Generating MSCHAPv2 server and client challenges as a function of the DH key agreement: in enforcing the dependence of the MSCHAP challenges on the DH exchange, a MitM is prevented from successfully establishing a secure tunnel with both the peer and legitimate server and succeed in obtaining the PAC credential.

* Cryptographic binding of EAP-FAST Phase 1 and the Phase 2 authentication method: by cryptographically binding key material generated in all methods, both peer and AS are assured that they were the sole participants of all transpired methods.

The binding of the MSCHAPv2 random challenge derivations to the DH key agreement protocol enables early detection of a MitM attack. This is required to guard from adversaries who may otherwise reflect the inner EAP authentication messages between the true peer and AS

and enforces that the adversary successfully respond with a valid challenge response.

The cryptographic binding is another reassurance that indeed the true peer and AS were the two parties ensuing both the tunnel establishment and inner EAP authentication conversations. While it would be sufficient to only support the cryptographic binding to mitigate the MitM; the extra precaution of binding the MSCHAP challenge to the DH key agreement affords the client earlier detection of a MitM and further guards the peer from having to respond to the success or failure of the adversaryÆs attempt to respond with a challenge response (e.g. indication of whether the adversary succeeded in breaking the peerÆs identity and password).

A failure in either step, results in no PAC provisioning. EAP-FAST invocation of provisioning using an unauthenticated tunnel can invoke certain procedures to guard implementations for potential MitM attacks. Detectors can be devised to warn the user when the peer encounters error conditions that warrant the likelihood of a MitM. For example, when the MSCHAPv2 server challenge response is never received or fails, the peer implementation can impose policy decisions to warn the user and respond to the likelihood that the failure was due to a MitM attack.

Similarly, to guard against attacks in the EAP-FAST Authentication that may force a peer to invoke in-band provisioning, guards and detectors can and should be implemented as part of the EAP-FAST Authentication protocols.

5.4 PAC Validation and User Credentials

In provisioning, the AS presents the peer with information such as a PAC-Key, PAC-Opaque and PAC-Info attributes. The peer must securely cache the PAC-Key and the PAC-Opaque which is bound to the A-ID provided as a PAC-Info attribute.

5.5 Generation of Diffie-Hellman Groups

The security of the DH key exchange is based on the difficulty of solving the Discrete Logarithm Problem (DLP). As algorithms and adversaries become more efficient in their abilities to precompute values for a given fixed group, it becomes more important for a

server to generate new groups as a means to allay this threat. The server could, for instance, constantly compute new groups in the background. Such an example is cited in [SECSH-DH].

Thus, the server can maintain a list of safe primes and corresponding generators to choose from. A prime p is safe, if:
p = 2q + 1 and q is prime

New primes may be generated in the background.

Initial implementations of the EAP-FAST provisioning exchange limit the generator to be 2 as it both improves the multiplication efficiency and still covers half of the space of possible residues. Furthermore, as the server defines the group used for the DH exchange, it may restrict the prime size to be 1024 bits.

Additionally, since the EAP-FAST provisioning exchange employs DH per [RFC 3268] to generate AES keys, the DH keys must provide enough entropy to ensure that a strong 128bit results from the DH key agreement.

EAP-FAST employs the 2048 bit DH groups defined in [RFC 3526].

5.6 PAC Storage Considerations

The main premise behind EAP-FAST is to protect the authentication stream over the media link. However, physical security is still an issue. Some care should be taken to protect the PAC on both the peer and server. The peer must store securely both the PAC-Key and PAC-Opaque, while the server must secure storage of its security association context used to consume the PAC-Opaque. Additionally, if manual provisioning is employed, the transportation mechanism used to distribute the PAC must also be secured.

Most of the attacks described here would require some level of effort to execute; conceivably greater than their value. The main focus therefore, should be to ensure that proper protections are used on both the client and server. There are a number of potential attacks which can be considered against secure key storage such as:

* weak passphrases
  On the client side, keys are usually protected by a passphrase. On some environments, this passphrase may be associated with the user's password. In either case, if an attacker can obtain the encrypted key for a range of users, he may be able to successfully attack a weak passphrase. The tools are already in place today to allow an attacker to easily attack all Outlook or Outlook Express users in an enterprise environment. Most viruses or worms of this sort attract attention to themselves by their action, but that need not be the case. A simple, genuine appearing email could surreptitiously access keys from known locations and email them directly to the attacker, attracting little notice.

* key finding attacks
  Key finding attacks are usually mentioned in reference to web servers, where the private SSL key may be stored securely, but at some point it must be decrypted and stored in system memory. An attacker with access to system memory can actually find the key by identifying their mathematical properties. To date, this attack appears to be purely theoretical and primarily acts to argue strongly for secure access controls on the server itself to prevent such unauthorized code from executing.

* key duplication , key substitution, key modification
  Once keys are accessible to an attacker on either the client or server, they fall under three forms of attack: key duplication, key substitution and key modification. The first option would be the most common, allowing the attacker to masquerade as the user in question. The second option could have some use if an attacker could implement it on the server. Alternatively, an attacker could use one of the latter two attacks on either the client or server to force a PAC re-key, and take advantage of the MitM/dictionary attack weakness of the EAP-FAST provisioning protocol.

Another consideration is the use of secure mechanisms afforded by the particular device. For instance, some laptops enable secure key storage through a special chip. It would be worthwhile for implementations to explore the use of such a mechanism.

6. IANA Considerations

This section explains the criteria to be used by the IANA for assignment of PAC TLV attribute and PAC-Type values. The

"Specification Required" policy is used here with the meaning defined in BCP 26 [RFC2434].

7. References

7.1 Normative

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[EAP] Blunk, L., et. al., "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC3268] Chown, P., "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)", RFC 3268, June 2002.

[RFC2119] Bradner, S., "Key words for use in RFCs to indicate Requirement Levels", RFC 2119, March 1997.

[RFC3546] Blake-Wilson, S., et al., "Transport Layer Security (TLS) Extensions", RFC 3546, June 2003.

[EAP-FAST] Cam-Winget, N., et al., "EAP Flexible Authentication via Secure Tunneling (EAP-FAST) ", draft-cam-winget-eap-fast-02 (work in progress), April 2005.

[TICKET] Salowey, J., et al, "TLS Session Resumption without Server-Side State", draft-salowey-tls-ticket-02.txt, February 2005

[MSCHAPv2] Zorn, G., ôMicrosoft PPP CHAP Extensions, Version 2ö, RFC 2759, January 2000.

[RFC2315] Kaliski, B., ôPKCS #7: Cryptographic Message Syntax Version 1.5ö, RFC 2315, March 1998.

7.2 Informative

[RFC2434] Narten, T., and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 2434, October 1998.

[RFC2631] Rescorla, E., "Diffie-Hellman Key Agreement Method", RFC 2631, January 1999.

[RFC3526] Kivinen, T., "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", RFC 3526, May 2003

[MITM] Puthenkulam, J., "The Compound Authentication Binding Problem", draft-puthenkulam-eap-binding-04 (expired), October 2003.

[RFC2486BIS]Aboba, et. al., "The Network Access Identifier", draft-ietf-radext-rfc2486bis-06.txt (work in progress), February, 2005.

8. Acknowledgments

The EAP-FAST design and protocol specification is based on the ideas and contributions from Pad Jakkahalli, Mark Krischer, Doug Smith, Ilan Frenkel and Jeremy Steiglitz of Cisco Systems, Inc.

9. Author's Addresses

Nancy Cam-Winget
Cisco Systems
3625 Cisco Way
San Jose, CA 95134
US
Phone: +1 408 853 0532
E-mail: ncamwing@cisco.com

David McGrew
Cisco Systems
San Jose, CA 95134
US
E-mail: mcgrew@cisco.com

Joseph Salowey
Cisco Systems
2901 3rd Ave
Seattle, WA 98121
US
Phone: +1 206 256 3380
E-mail: jsalowey@cisco.com

Hao Zhou
Cisco Systems
4125 Highlander Parkway
Richfield, OH 44286
US
Phone : +1 330 523 2132
E-mail: hzhou@cisco.com

10. Appendix: Examples

10.1 Example 1: Successful Tunnel PAC Provisioning

The following exchanges show anonymous DH with a successful EAP-MSCHAPv2 exchange within Phase 2 to provision a Tunnel PAC, the conversation will appear as follows:

```
    Authenticating Peer     Authenticator
    -------------------     -------------
                            <- EAP-Request/
                            Identity
    EAP-Response/
    Identity (MyID1) ->
                            <- EAP-Request/
                            EAP-Type=EAP-FAST, V=1
                            (EAP-FAST Start, S bit set, A-ID)

    EAP-Response/
    EAP-Type=EAP-FAST, V=1
    (TLS client_hello without
    PAC-Opaque extension)->
                            <- EAP-Request/
                            EAP-Type=EAP-FAST, V=1
                            (TLS server_hello,
                             TLS Server Key Exchange
                             TLS Server Hello Done)
```

```
       EAP-Response/
       EAP-Type=EAP-FAST, V=1 ->
       (TLS Client Key Exchange
        TLS change_cipher_spec,
        TLS finished)

                               <- EAP-Request/
                               EAP-Type=EAP-FAST, V=1
                               (TLS change_cipher_spec
                                TLS finished)
       EAP-Response/
       EAP-Type=EAP-FAST, V=1 ->
       (Acknowledgement)

       TLS channel established
       (messages sent within the TLS channel)

                              <-  EAP Payload TLV,
                                  EAP-Request/
                                  EAP Identity Request

       EAP Payload TLV, EAP-Response/
       EAP Identity Response ->

                              <-  EAP Payload TLV,
                                  EAP-Request,
                                  EAP-MSCHAPV2, Challenge

       EAP Payload TLV, EAP-Response,
       EAP-MSCHAPV2, Response) ->

                              <-  EAP Payload TLV,
                                  EAP-Request, MSCHAPV2, Success)
       EAP Payload TLV, EAP-Response,
       MSCHAPV2, Success) ->
                               <- Intermediate Result TLV (Success)
                                  Binding-TLV=(Version=1,SNonce,
                                  CompoundMAC)

       Intermediate Result TLV (Success)
       Binging-TLV=(Version=1,
       CNonce, CompoundMAC)
                               <- Result TLV (Success)
                                  PAC TLV
```

```
      Result TLV (Success)
      PAC Acknowledgment ->

      TLS channel torn down
      (messages sent in cleartext)

                              <- EAP-Success
```

10.2 Example 2: Successful Tunnel PAC Provisioning with Password Change

The following exchanges show where EAP-MSCHAPv2 with password change within Phase 2 to provision a Tunnel PAC, the conversation will appear as follows:

```
      Authenticating Peer     Authenticator
      -------------------     -------------
                              <- EAP-Request/
                              Identity
      EAP-Response/
      Identity (MyID1) ->
                              <- EAP-Request/
                              EAP-Type=EAP-FAST, V=1
                              (EAP-FAST Start, S bit set, A-ID)

      EAP-Response/
      EAP-Type=EAP-FAST, V=1
      (TLS client_hello without
      PAC-Opaque extension)->
                              <- EAP-Request/
                              EAP-Type=EAP-FAST, V=1
                              (TLS Server Key Exchange
                               TLS Server Hello Done)

      EAP-Response/
      EAP-Type=EAP-FAST, V=1 ->
      (TLS Client Key Exchange
       TLS change_cipher_spec,
       TLS finished)

                              <- EAP-Request/
                              EAP-Type=EAP-FAST, V=1
                              (TLS change_cipher_spec
                               TLS finished)
```

```
       EAP-Response/
       EAP-Type=EAP-FAST, V=1 ->
       (Acknowledgement)

       TLS channel established
       (messages sent within the TLS channel)

                               <-  EAP Payload TLV
                                   EAP-Request/
                                   EAP Identity Request

       EAP Payload TLV
       EAP-Response/
       EAP Identity Response ->


                               <- EAP-Payload TLV
                                  EAP-Request/
                                  EAP-MSCHAPV2, Challenge

       EAP Payload TLV, EAP-Response,
       EAP-MSCHAPV2, Response) ->

                               <- EAP Payload TLV,
                                  EAP-Request, MSCHAPV2, Failure,
                                  Error Code =
                                  ERROR_PASSWD_EXPIRED (E=648))
       EAP Payload TLV, EAP-Response,
       MSCHAPV2, Change Password Response) ->

                               <-  EAP Payload TLV, EAP-Request,
                                   MSCHAPV2, Success)

       EAP Payload TLV, EAP-Response,
       MSCHAPV2, Success) ->
                                <- Intermediate Result TLV (Success)
                                   Binding-TLV=(Version=1,SNonce,
                                   CompoundMAC)

       Intermediate Result TLV (Success)
       Binging-TLV=(Version=1,
       CNonce, CompoundMAC)
                                <- Result TLV (Success)
                                   PAC TLV
```

```
        Result TLV (Success)
        PAC Acknowledgement  ->

        TLS channel torn down
        (messages sent in cleartext)

                                <- EAP-Success
```

10.3 Example 3: Failed Provisioning

The following exchanges show a failed EAP-MSCHAPV2 exchange within Phase 2, where the peer failed to authenticate the Server. The conversation will appear as follows:

```
        Authenticating Peer     Authenticator
        -------------------     -------------
                                <- EAP-Request/
                                Identity
        EAP-Response/
        Identity (MyID1) ->
                                <- EAP-Request/
                                EAP-Type=EAP-FAST, V=1
                                (EAP-FAST Start, S bit set, A-ID)

        EAP-Response/
        EAP-Type=EAP-FAST, V=1
        (TLS client_hello without
        PAC-Opaque extension)->
                                <- EAP-Request/
                                EAP-Type=EAP-FAST, V=1
                                (TLS Server Key Exchange
                                 TLS Server Hello Done)
        EAP-Response/
        EAP-Type=EAP-FAST, V=1 ->
        (TLS Client Key Exchange
         TLS change_cipher_spec,
         TLS finished)

                                <- EAP-Request/
                                EAP-Type=EAP-FAST, V=1
                                (TLS change_cipher_spec
```

```
                                TLS finished)
        EAP-Response/
        EAP-Type=EAP-FAST, V=1 ->
        (Acknowledgement)

        TLS channel established
        (messages sent within the TLS channel)

                                <- EAP Payload TLV
                                   EAP-Request/EAP Identity Request

        Eap Payload TLV
        EAP-Response/
        EAP Identity Response ->

                                <-  EAP Payload TLV, EAP-Request,
                                    EAP-MSCHAPV2, Challenge

        EAP Payload TLV, EAP-Response,
        EAP-MSCHAPV2, Response ->

                                <-  EAP Payload TLV, EAP-Request,
                                    EAP-MSCHAPV2, Success)

        EAP Payload TLV, EAP-Response,
        EAP-MSCHAPV2, Failure) ->

                                <-  Result TLV (Failure)

        Result TLV (Failure) ->

        TLS channel torn down
        (messages sent in cleartext)

                                 <- EAP-Failure
```

10.4 Example 4: Provisioning a Authentication ServerÆs Trusted Root Certificate

The following exchanges show a successful provisioning of a server trusted root certificate using anonymous DH and EAP-MSCHAPV2 exchange within Phase 2, the conversation will appear as follows:

```
   Authenticating Peer     Authenticator
   -------------------     -------------
                           <- EAP-Request/
                           Identity
   EAP-Response/
   Identity (MyID1) ->
                           <- EAP-Request/
                           EAP-Type=EAP-FAST, V=1
                           (EAP-FAST Start, S bit set, A-ID)

   EAP-Response/
   EAP-Type=EAP-FAST, V=1
   (TLS client_hello without
   PAC-Opaque extension)->
                           <- EAP-Request/
                           EAP-Type=EAP-FAST, V=1
                           (TLS server_hello,
                           (TLS Server Key Exchange
                            TLS Server Hello Done)

   EAP-Response/
   EAP-Type=EAP-FAST, V=1 ->
   (TLS Client Key Exchange
    TLS change_cipher_spec,
    TLS finished)

                           <- EAP-Request/
                           EAP-Type=EAP-FAST, V=1
                           (TLS change_cipher_spec
                            TLS finished)
                            EAP-Payload-TLV[
                            EAP-Request/Identity])

   // TLS channel established
      (messages sent within the TLS channel)

   // First EAP Payload TLV is piggybacked to the TLS Finished as
      Application Data and protected by the TLS tunnel

   EAP-Payload TLV/
   [EAP Identity Response] ->

                          <-  EAP Payload TLV, EAP-Request,
```

```
                              [EAP-MSCHAPV2, Challenge]

   EAP Payload TLV, EAP-Response,
   [EAP-MSCHAPV2, Response] ->

                           <-  EAP Payload TLV, EAP-Request,
                               [EAP-MSCHAPV2, Success Request]

   EAP Payload TLV, EAP-Response,
   [EAP-MSCHAPV2, Success Response] ->

                            <- Crypto-Binding TLV (Version=1,
                               EAP-FAST Version=1, Nonce,
                               CompoundMAC),

   Crypto-Binding TLV (Version=1
   EAP-FAST Version=1, Nonce,
   CompoundMAC)
   Server-Trusted-Root TLV
   [Type = PKCS#7 ] ->
                            <- Result TLV (Success)
                               Server-Trusted-Root TLV
                               [PKCS#7 TLV]

   Result TLV (Success) ->

   // TLS channel torn down
      (messages sent in cleartext)

                            <- EAP-Failure
```

10.5 Example 5: Provisioning a User Authorization PAC

The following exchanges demonstrate how a User Authorization PAC is provisioned in Phase 2. The conversation will appear as follows:

```
   Authenticating Peer     Authenticator
   -------------------     -------------
                           <- EAP-Request/
                           Identity
   EAP-Response/
   Identity (MyID1) ->
                           <- EAP-Request/
```

```
                          EAP-Type=EAP-FAST, V=1
                          (EAP-FAST Start, S bit set, A-ID)

   EAP-Response/
   EAP-Type=EAP-FAST, V=1
   (TLS client_hello with
   PAC-Opaque extension)->
                          <- EAP-Request/
                          EAP-Type=EAP-FAST, V=1
                          (TLS server_hello,
                          (TLS change_cipher_spec,
                           TLS finished)
   EAP-Response/
   EAP-Type=EAP-FAST, V=1 ->
   (TLS change_cipher_spec,
    TLS finished)

   TLS channel established
   (messages sent within the TLS channel)

                         <-  EAP-Payload TLV, EAP-Request,
                             EAP-GTC, Challenge

   EAP-Payload TLV, EAP-Response,
   EAP-GTC, Response with both
   user name and password) ->

   optional additional exchanges (new pin mode,
   password change etc.) ...

                          <- Crypto-Binding TLV=(Version=1,
                             EAP-FAST Version =1, Nonce,
                             CompoundMAC)
                             Result TLV (Success)

   Crypto-Binding TLV=(Version=1,
   EAP-FAST Version=1, Nonce,
   CompoundMAC)
   Result TLV (Success)
   Request-Action TLV
   (Action=1-Process TLV)
   PAC TLV with PAC-Type=User Authorization PAC)->

                          <- Result TLV (Success)
```

```
                              PAC TLV with User Authorization
                              PAC

   Result TLV (Success) ->

   TLS channel torn down
   (messages sent in cleartext)

                            <- EAP-Success
```

11. Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

12. Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE

INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

13. Copyright Statement

14. Expiration Date

This memo is filed as <draft-cam-winget-eap-fast-provisioning-00.txt>, and expires January 17, 2006.